

On Classes of Distributed Petri Nets

Von der
Carl-Friedrich-Gauß-Fakultät
der Technischen Universität Carolo-Wilhelmina zu Braunschweig

zur Erlangung des Grades eines
Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigte Dissertation

von
Jens-Wolfhard Schicke-Uffmann
geboren am 3. Januar 1987
in Braunschweig

Eingereicht am: 19. September 2017

Disputation am: 11. April 2018

1. Referentin: Prof. Dr.-Ing. Ina Schaefer
2. Referent: Prof. Dr. Rob van Glabbeek
3. Referent: Prof. Dr.-Ing. Uwe Nestmann

Acknowledgements

This work would not have happened without the continuous encouragement by Prof. Dr. Ursula Goltz and Prof. Dr. Rob van Glabbeek, from whom I had the fortune to learn many things, some important ones of which relate only tangentially to this thesis.

During my work I enjoyed the collegueship of Dr. Kirstin Peters and Stephan Mennicke, with whom I had many fruitful discussions, arguments and also papers. Furthermore I must thank Prof. Dr.-Ing. Uwe Nestmann, in particular for his ability to eliminate hand-wavy arguments from many of our discussions and paper drafts.

My thanks include Sylvia Poßenau, Annette Thurow and Elke Rock who also reviewed drafts of this thesis. Similarly, many anonymous reviewers made valuable suggestions to the underlying papers or plain out rejected them when not quite good enough yet. All remaining errors are obviously mine.

I am very glad that Prof. Dr.-Ing. Ina Schaefer and Prof. Dr. Wolf-Tilo Balke were willing to fill the necessary roles on my PhD committee, after Prof. Dr. Goltz became unable to do so. I could not wish for a more impartial evaluation of what I submit as my thesis.

Thanks also go to Norman Megill, the author of Metamath, which finally taught me how to do formal proofs correctly. Without it, some proofs reworked for this thesis would probably be wrong, but shorter.

The Deutsche Forschungsgemeinschaft, the Deutscher Akademischer Austauschdienst and the National ICT Australia funded parts of my research, the TU Braunschweig the huge remaining part.

Contents

<u>1</u>	<u>Introduction</u>	7
<u>2</u>	<u>Basic Notions</u>	11
2.1	<u>Petri Nets</u>	12
2.2	<u>Behavioural Equivalences</u>	15
<u>3</u>	<u>Distributed Nets</u>	27
3.1	<u>Asynchronous Net Classes</u>	27
3.2	<u>The Asynchronous Implementation</u>	34
3.3	<u>An Alternate Characterisation</u>	38
3.3.1	<u>Distributed Nets</u>	39
3.3.2	<u>LSGA Nets</u>	40
3.3.3	<u>Relation between LSGA Nets and Distributed Nets</u>	42
<u>4</u>	<u>N-Free Nets and Related Classes</u>	47
4.1	<u>Free-Choice and Extended Free-Choice</u>	47
4.2	<u>Behavioural Free-Choice</u>	51
4.2.1	<u>BFC-nets without Self-Loops</u>	53
4.2.2	<u>BFC-nets and Asymmetric Choice</u>	54
4.3	<u>Symmetric Asynchrony</u>	57
4.4	<u>Short Overview of the Results So Far</u>	58
<u>5</u>	<u>M-Free Nets and Related Classes</u>	61
5.1	<u>Reachable-M Free</u>	61
5.1.1	<u>Characterising Distributability</u>	63
5.1.2	<u>Nets with Reversible Transitions</u>	65
5.1.3	<u>The Conflict Replicating Implementation</u>	68
5.2	<u>Proving Implementations Correct</u>	75
5.3	<u>The Correctness Proof</u>	87
<u>6</u>	<u>M-Containing Nets</u>	107
6.1	<u>Instability of Ms</u>	107
6.2	<u>Stability of Ms</u>	107
6.3	<u>Infinite Number of Less Restrictive Classes</u>	112

<u>7 Distributed Net Implementation</u>	117
7.1 <u>Petri Net Markup Language</u>	117
7.2 <u>Architecture Overview</u>	118
7.3 <u>Implementation Details and Networking</u>	120
7.4 <u>Performance and Comparison</u>	121
<u>8 Context, Discussion and Conclusion</u>	125
8.1 <u>Ways to Evade the (Negative) Theorems</u>	125
8.1.1 <u>Probabilistic Algorithms</u>	125
8.1.2 <u>Time and Bounded Message Delay</u>	126
8.1.3 <u>Quantum Mechanics</u>	127
8.2 <u>Related Work</u>	128
8.2.1 <u>Petri Nets</u>	128
8.2.2 <u>Process Algebra</u>	130
8.2.3 <u>Applications</u>	132
8.3 <u>Conclusions and Open Questions</u>	133
<u>9 Literature</u>	137
<u>10 Deutschsprachige Zusammenfassung</u>	147
<u>A Experimental Data</u>	151

Chapter 1

Introduction

Few computer systems exist in isolation. Most interact with an environment on a regular and continuous basis. A computer without any interaction would be pretty useless in most contexts. At the same time many systems can be decomposed into smaller components which could rightfully be considered computers in their own right. This is certainly true for the vast networks of interdependent web services offering social networks to billions of users, but remains so down to the scale of single cars, smartphones and medical implants, to give just some examples.

It has long been recognised that distributed systems are hard to design and implement. Many of the problems stem from the inability even of excellent programmers to correctly survey all possible execution schedules within such a system, hence overlooking subtle bugs which will emerge only under few schedules and are very hard to understand. More fundamental problems however occur when the system is required to present a consistent façade to a – typically spatially – distributed set of users or needs to control multiple parts of the environment coupled through other channels than only those of the control system. In such contexts, the propagation delay of information gathered from the environment throughout the system can become relevant. This thesis is concerned with this latter type of problems only, trying to delineate what can conceptually be provided by a distributed system, what cannot, and why.

A practical example without computers will elucidate the fundamental problem quite well: This evening, you wish to collaborate with a friend of yours on a hard part of your joint paper. However, this will only be possible if you manage to call each other, so you can finally think about a solution together. Unfortunately, recent revelations forced you to use some cryptographically secured, but still slightly experimental phone software which might or might not work for each of you in any given moment. Alternatively, each of you could just call it a day and invest your time less scientifically, say by taking a nice walk in the park.

Being the scientifically minded person that you are, you try to call your friend, only to find out that indeed the phone software is currently not working.

So the park it is. Just as you fetch your shoes however, you notice that it has started raining. Another attempted call to your friend is not answered, possibly because she has already given up and has gone for a walk herself. Or maybe she's just ditching the shoes again because there is rain, too, and will be calling any minute?

Clearly it would be all easier if you would not need to run between door and computer to check for rain and working software respectively. While your laptop is booting and you are carrying it to the door, you wonder how long you will let the phone software ring before giving up this time. As you contemplate what to do if it stops raining during those seconds, you find that it would have been a good idea to agree on a better protocol beforehand. . .¹

The hard problem here is to synchronise two parties whose available actions change over time. If communication is faster than environmental change, everyone can just tell what they *can* currently do and people can find a consensus² on how to proceed. If however communication is slower than changes in environment, such strategies are not guaranteed to work – and will fail in case of not just stochastic but actively antagonistic environments.

The question then is: What kind of system behaviour will still succeed, even in the face of slow communication and a fast-changing environment.

To answer that question, this thesis starts with a formalisation of distributed systems and ways to compare their observable behaviour. The main part then classifies synchronous systems or specifications by the possibility to find distributed systems which behave comparably to them. On some of the cases where we show that no behaviourally equivalent distributed system to a certain specification exists we also show that our negative example is minimal in the sense that it is a necessary part of all those systems which can not be distributed. Afterwards, an executable, if prototypical, implementation of some of our constructions is given. The thesis then finishes with the usual discussion of related work and the broader context.

This thesis is largely comprised of papers published during the years 2008 to 2014, as stated at the beginning of chapters and sections where applicable. I unified definitions and notation between all papers, to make the connections between them more formally accessible. Naturally, recurring parts have also been extracted, as the value of repeated introductions of, say, multisets seems limited. Section 6.3, Chapter 7, and Section 8.1 contain entirely new material, the latter two concerning practical applicability of the theoretical work.

My first exposure to the topic of this thesis happened during my Studienarbeit [Sch08] parts of which ended up in a joint paper [GGS09] with Ursula Goltz and Rob van Glabbeek. Based upon these initial results, we started to isolate irreducibly synchronous system behaviour. To allow more thorough research

¹Just as you realise that nature will not switch rain on and off arbitrarily fast and open the door, a roof avalanche occurs.

²Which of course carries its own problems. Those of breaking symmetry (which are just slightly out of scope of this thesis) and those computers don't have to deal with.

in the area, and to reduce the probability of chasing model-specific artefacts, we connected with Uwe Nestmann and Kirstin Peters who followed a parallel research program based upon the π -calculus instead of Petri nets at TU Berlin. A joint grant proposal to the German Science Foundation was granted in 2010 and provided funding for the next two years. Later, Stephan Mennicke joined us at TU Braunschweig and concentrated on characterising system behaviour using stronger notions of asynchrony. In 2013 the German Science Foundation accepted our follow-up proposal and the project continued.

Chapter 2

Basic Notions

The presentation in this chapter is partially taken from [\[GGSU13\]](#).

Definition 1 Let X be a set.

1. The *identity function* on X , $\text{Id}_X : X \rightarrow X$ is defined by $\text{Id}_X(x) := x$.
2. A *signed multiset* over X is an element $A \in \mathbb{Z}^X$, i. e. a function $A : X \rightarrow \mathbb{Z}$.

Let A, B be signed multisets over X .

3. A is a *multiset* iff $A(X) \subseteq \mathbb{N}$.
4. $x \in X$ is an *element* of A , notation $x \in A$, iff $A(x) \neq 0$.
5. A is *finite* iff the set $\{x \mid x \in A(x)\}$ is finite.
6. $A \leq B$ iff $\forall x \in X. A(x) \leq B(x)$.
7. $(A \cup B)(x) := \max(A(x), B(x))$ and $(A \cap B)(x) := \min(A(x), B(x))$ denote signed multisets over X .
8. Similarly $(A + B)(x) := A(x) + B(x)$, $(A - B)(x) := A(x) - B(x)$, and $(k \cdot A)(x) := k \cdot A(x)$ with $k \in \mathbb{Z}$ denote signed multisets over X .
9. Any function $f : X \rightarrow \mathbb{Z}$ or $f : X \rightarrow \mathbb{Z}^Y$ from X to either the integers or the signed multisets over some set Y extends to the finite signed multisets A over X by $f(A) = \sum_{x \in X} A(x) \cdot f(x)$.
10. Two signed multisets $A : \mathbb{Z}^X$ and $C : \mathbb{Z}^Y$ are *extensionally equivalent* iff $x \in X \cap Y \Rightarrow A(x) = C(x)$, $x \in Y \setminus X \Rightarrow C(x) = 0$, and $x \in X \setminus Y \Rightarrow A(x) = 0$.

Extensionally equivalent signed multisets are not distinguished for the remainder of this thesis. A multiset A with $\forall x \in A. A(x) \leq 1$ is identified with the set $\{x \mid A(x) = 1\}$. (This also defines signed multiset \emptyset .)

2.1 Petri Nets

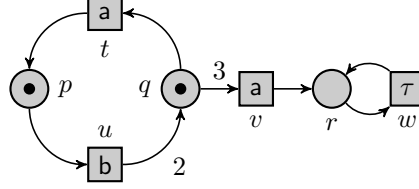


Figure 2.1: An example Petri net

A well-known abstraction from distributed systems is the model of Petri nets, unifying the aspects of causally dependent and concurrent actions in a neat graphical representation. Petri nets stem from the ideas given in the PhD thesis of Carl Adam Petri [Pet62]. He forced system states to be structured by means of local states, capturing concepts of distributed systems, where components are distributed over several locations. That actions of a single component depend only on local state information is represented naturally by the firing rule of Petri nets. In addition, Petri nets offer an intuitive notion of concurrent computation by the *step firing rule*, a generalization of the firing rule allowing more than one transition to fire in parallel (or unordered). Hence distributed systems specified by Petri nets can be analysed while fully taking into account any uncertainties in the order in which actions occur. Although decidable, many interesting properties such as *liveness* or *boundedness* of Petri nets are hard in terms of computational complexity [Esp98]. While Petri nets have been extended in a myriad of ways to fit different application domains, this thesis is concerned only with very basic features.

Definition 2 Let Act be a set of *actions*, and let $\tau \notin \text{Act}$ be the *invisible action*. A *Petri net* is a tuple $N = (S, T, F, M_0, \ell)$ with

- S a set of *places*,
- T a set of *transitions*, $S \cap T = \emptyset$,
- $F : (S \times T \cup T \times S) \rightarrow \mathbb{N}$ a *flow relation* including *arc weights*,
- $M_0 : S \rightarrow \mathbb{N}$ an *initial marking*, and
- $\ell : T \rightarrow \text{Act} \cup \{\tau\}$ a *labelling function*.

To save redundant words¹ we will from now on just say *net* instead of Petri net.

Nets are depicted by drawing the places as circles and the transitions as boxes, containing their label. Identities of places and transitions are displayed next to the net element. When $F(x, y) > 0$ for $x, y \in S \cup T$ there is an arrow

¹about a page of “Petri” across the remaining thesis

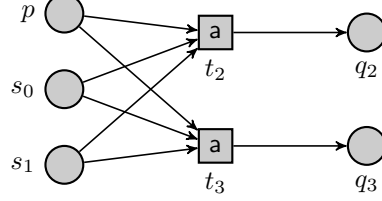
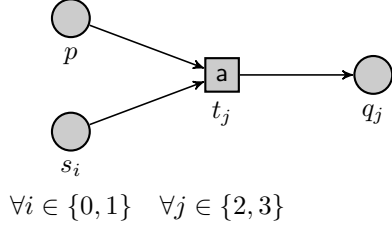


Figure 2.2: A net with quantifiers.

Figure 2.3: The same net expanded.

(*arc*) from x to y , labelled with the *arc weight* $F(x, y)$. Weights of 1 are omitted. When a net represents a concurrent system, a global state of this system is given as a *marking*, a multiset M of places, depicted by placing $M(s)$ dots (*tokens*) in each place s . The initial state is M_0 .

To compress the graphical notation, we allow universal quantifiers of the form $\forall x.\phi(x)$ to appear in the drawing (cf. Figures 2.2 and 2.3). A quantifier replaces occurrences of x in place and transition identities with all concrete values for which $\phi(x)$ holds, possibly creating a set of places, respectively transitions, instead of the depicted single one. Accordingly, an arc of which only one end is replicated by a given quantifier results in a fan of arcs, one for each replicated element. If both ends of an arc are affected by the same quantifier, an arc is created between pairs of elements corresponding to the same x , but not between elements created due to differing values of x .

Definition 3 Let $N = (S, T, F, M_0, \ell)$ be a net, and let $x \in S \cup T$.

The multisets $\bullet x, x^\bullet$ are given by $(\bullet x)(y) := F(y, x)$ and $(x^\bullet)(y) := F(x, y)$ respectively. If $x \in T$ the elements of $\bullet x$ are *preplaces*, those of x^\bullet *postplaces*. If $x \in S$ they are *pretransitions* and *posttransitions* respectively. The *token replacement function* $\llbracket _ \rrbracket : T \rightarrow \mathbb{Z}^S$ is defined as $\llbracket t \rrbracket := t^\bullet - \bullet t$. These functions extend to signed multisets as usual (see Definition 1).

The behaviour of a net is defined by the possible moves between markings M and M' , which take place when a finite multiset G of transitions *fires*. In that case, each occurrence of a transition t in G consumes $F(s, t)$ tokens from each place s . Naturally, this can happen only if M makes all these tokens available in the first place. Next, each t produces $F(t, s)$ tokens in each s .

Definition 4 Let $N = (S, T, F, M_0, \ell)$ be a net. A multiset $M \in \mathbb{N}^S$ is a *marking of N* .

Let M, M' be two markings of N . A non-empty, finite multiset $G \in \mathbb{N}^T$ is a *step from M to M'* , written $M [G]_N M'$, iff

1. $M \geq \bullet G$, and
2. $M' = M - \bullet G + G^\bullet = M + \llbracket G \rrbracket$.

The step is often also said to *fire* in M , resulting in a marking M' . The set $[M_0]_N$ of *reachable markings of N* is defined as the smallest set containing M_0 that is closed under $[G]_N$, meaning that if $M \in [M_0]_N$ and $M [G]_N M'$ for any G then $M' \in [M_0]_N$. For explicit G , we can also omit the redundant set-notation, e. g. $M[t, u]_N M'$ iff $M[\{t, u\}]_N M'$.

Definition 5 Let $N = (S, T, F, M_0, \ell)$ be a net and $t, u \in T$.

The concurrency relation is defined as

$$t \smile u \text{ iff } t \neq u \wedge \exists M \in [M_0]_N. M[\{t, u\}]$$

Definition 6 Let $N = (S, T, F, M_0, \ell)$ be a net. Let M, M' be two markings of N , $a \in \text{Act}$, $A : \mathbb{N}^{\text{Act}}$, $a_0 a_1 a_2 \dots a_n = \sigma \in \text{Act}^*$, and $A_0 A_1 A_2 \dots A_n = \rho \in (\mathbb{N}^{\text{Act}})^*$.

1. $M \xrightarrow{a}_N M'$ iff $\exists t \in T. M[\{t\}]_N M' \wedge \ell(t) = a$,
2. $M \xrightarrow{\tau}_N M'$ iff $\exists t \in T. M[\{t\}]_N M' \wedge \ell(t) = \tau$,
3. $M \xrightarrow{\tau}_N^\infty$ iff $\forall n \in \mathbb{N} \exists M'. M \xrightarrow{\tau}^n M'$, ²
4. $M \xrightarrow{A}_N M'$ iff $\exists G \in \mathbb{N}^T. M[G]_N M' \wedge \ell(G) = A$,
5. $M \xRightarrow{\sigma}_N M'$ iff $M \xrightarrow{\tau}_N^* \xrightarrow{a_0}_N \xrightarrow{\tau}_N^* \xrightarrow{a_1}_N \xrightarrow{\tau}_N^* \xrightarrow{a_2}_N \dots \xrightarrow{a_n}_N \xrightarrow{\tau}_N^* M'$
6. $M \xRightarrow{\rho}_N M'$ iff $M \xrightarrow{\tau}_N^* \xrightarrow{A_0}_N \xrightarrow{\tau}_N^* \xrightarrow{A_1}_N \xrightarrow{\tau}_N^* \xrightarrow{A_2}_N \dots \xrightarrow{A_n}_N \xrightarrow{\tau}_N^* M'$

The negations of these relations are denoted as usual, i. e. by $\not\xrightarrow{a}_N$, $\not\xrightarrow{\tau}_N$, and $\not\xrightarrow{A}_N$ respectively. When omitting the right (target) marking, it is assumed to be existentially quantified, e. g. $M \xrightarrow{a}_N$ iff $\exists M'. M \xrightarrow{a}_N M'$. The subscript N is omitted if clear from context.

The following Definition 7 gives some restrictions on nets which this thesis employs in many places. A particularly easy to analyse subclass is of course that of finite nets. Note that even though a net is finite, it might still come with an infinite state space. Nearly as easy are finitary nets, where at all times only finitely many tokens exist and only finitely many transitions are enabled.

The mapping from net transitions to labels can help with modelling a system which needs to perform the same external action in different internal states. When analysing the expressive power of nets however, the additional layer of indirection mostly gets in the way: With labels, the impossibility theorems presented later would need additional clauses requiring distinctness of labels, to exclude spurious examples like nets where all transitions are labelled identically.

Similarly, the effects we are interested in already show themselves when considering only nets which are 1-safe, i. e. only allow one token to reside on a

²where $\xrightarrow{\tau}^n$ is the n-fold composition of $\xrightarrow{\tau}$ with itself

place at the same time. Our results however can be carried over without too much work to the slightly larger class of structural conflict nets, where multiple tokens are allowed on a single place as long as multiple transitions never compete for these tokens.

Definition 7 A net $N = (S, T, F, M_0, \ell)$ is called

1. *finite* iff S and T are finite sets,
2. *finitary* iff M_0 is finite, $\forall t \in T. \bullet t \neq \emptyset$ and x^\bullet is finite for all $x \in S \cup T$,
3. *unlabelled* iff $\ell = \text{Id}_T$,
4. *plain labelled* or *plain* iff $(\ell(t) = \ell(u)) \Rightarrow (\ell(t) = \tau \vee t = u)$,
5. *1-safe* iff $M \in [M_0] \Rightarrow M(S) \subseteq \{0, 1\}$.
6. a *structural conflict net* iff $M \in [M_0] \wedge M[\{t, u\}] \Rightarrow \bullet t \cap \bullet u = \emptyset$.

Lemma 1 Let $N = (S, T, F, M_0, \ell)$ be an unlabelled net. Let $\sigma \in \text{Act}^*$.
 $(M_0 \xrightarrow{\sigma} M_1 \wedge M_0 \xrightarrow{\sigma} M_2) \Rightarrow M_1 = M_2$.

Proof: $\sigma = a_1 a_2 \dots a_n$ and each a_i corresponds to the unique transition $t \in T$ with $\ell(t) = a_i$. By definition $M[\{t\}]M' \wedge M[\{t\}]M'' \Rightarrow M' = M''$ and the result follows by induction. \square

2.2 Behavioural Equivalences

When considering alternative possible implementations of an existing system or an abstract specification, one would like to define those implementations which are “correct”. Depending on application context, however, which deviations in behaviour are acceptable and which are not, differs. Hence various formalisations of being a “correct” implementation have been proposed. For an overview see [Gla01, Gla93].

All these formalisations take the form of an equivalence between behaviours. If the implementation behaviour is equivalent to the specification behaviour, it can be considered “correct”. Two such equivalences can be compared on whether they imply each other. If an equivalence \approx_a implies another \approx_b , then \approx_a is the finer of the two, it detects more differences of behaviour. Ordering the equivalences into a lattice this way, the degrees to which various features of behaviour are considered relevant by the equivalences gives rise to semi-independent dimensions along which equivalences get finer. A comprehensive presentation of the different equivalences and a visualisation of the resulting lattice can be found in [Gla93].

Definition 8 Let $N = (S, T, F, M_0, \ell)$ be a net.

A string $\sigma \in \text{Act}^*$ is a *weak trace* of N iff $M_0 \xrightarrow{\sigma}$.

Two nets are *weak trace equivalent* iff they have the same set of weak traces.

Definition 9 Let $N = (S, T, F, M_0, \ell)$ be a net.

A string $\sigma \in \text{Act}^*$ is a *completed weak trace* of N iff $M_0 \xRightarrow{\sigma} M$ for some M with $\forall a \in \text{Act}. M \not\xrightarrow{a}$ and $M \not\xrightarrow{\tau}$.

A string $\sigma \in \text{Act}^*$ is a *diverging weak trace* of N iff $M_0 \xRightarrow{\sigma} M$ for some M with $M \xrightarrow{\tau}^\infty$.

Two nets are *weak completed trace equivalent* iff they have the same weak traces, the same completed weak traces and the same diverging weak traces.

Definition 10 Let $N = (S, T, F, M_0, \ell)$ be a net.

A string of multisets $\sigma \in (\mathbb{N}^{\text{Act}})^*$ is a *weak step trace* of N iff $M_0 \xRightarrow{\sigma}$.

A string of multisets $\sigma \in (\mathbb{N}^{\text{Act}})^*$ is a *completed weak step trace* of N iff $M_0 \xRightarrow{\sigma} M$ for some M with $\forall a \in \text{Act}. M \not\xrightarrow{a}$ and $M \not\xrightarrow{\tau}$.

A string of multisets $\sigma \in (\mathbb{N}^{\text{Act}})^*$ is a *diverging weak step trace* of N iff $M_0 \xRightarrow{\sigma} M$ for some M with $M \xrightarrow{\tau}^\infty$.

Two nets are *weak completed step trace equivalent* iff they have the same weak step traces, the same completed weak traces and the same diverging weak step traces.

Definition 11 Let $N = (S, T, F, M_0, \ell)$ be a net.

A pair $(\sigma, X) \in \text{Act}^* \times 2^{\mathbb{N}^{\text{Act}}}$ is a *step failure pair* of N if $M_0 \xRightarrow{\sigma} M$ for some M with $M \not\xrightarrow{\tau}$ and $\forall A \in X. M \not\xrightarrow{A}$. It is a *finite step failure pair* if additionally X is finite. We write $\mathcal{F}(N)$ for the set of finite step failure pairs of N .

Two nets are *(finite) step failures equivalent* iff they have the same (finite) step failure pairs. We write $N_1 \approx_{\mathcal{F}} N_2$ iff N_1 and N_2 are finite step failures equivalent.

Definition 12 Let $N = (S, T, F, M_0, \ell)$ be a net.

A pair $(\sigma, X) \in \text{Act}^* \times 2^{\mathbb{N}^{\text{Act}}}$ is a *step ready pair* of N if $M_0 \xRightarrow{\sigma} M$ for some M with $M \not\xrightarrow{\tau}$ and $A \in X \Leftrightarrow M \xrightarrow{A}$.

Two nets are *step readiness equivalent* iff they have the same step ready pairs. We write $N_1 \approx_{\mathcal{R}} N_2$ iff N_1 and N_2 are step readiness equivalent.

The definitions of failures and readiness equivalence given above are based on the *stable* variants developed in [BKO87, Ros98] of the failures resp. readiness equivalence given in [BHR84, OH86] and extend them further by capturing steps instead of singleton actions.

If a system has the potential to engage in an infinite sequence of internal actions, one speaks of *divergence*. When comparing two systems without divergence, the stable finite failures equivalence coincides with the failures equivalence of [BHR84]. When comparing systems of which one is known to be divergence-free – as we will do in this thesis – stable failures semantics is strictly less discriminating than the failures equivalence of [BHR84] – only the latter guarantees that the other system is divergence-free as well. As a less discriminating

equivalence will give rise to stronger results about the absence of distributed implementations of certain systems, we use the stable variant. Similar considerations lead to the restriction that X be finite. As the equivalence is thereby made less discriminating, we obtain stronger impossibility results.

A variation given in [TV89] has σ containing steps instead of singleton actions. Again, this leads to a finer equivalence and weaker negative results.

Another equivalence notion we employ is based on the *weak bisimilarity* of [Mil89], which is a bit less discriminating than branching bisimilarity as proposed in [GW89]. (Branching) bisimilarity *with explicit divergence* [GW96, GLT09], is a variant of (branching) bisimilarity that fully respects the diverging behaviour of related systems. Since here we only compare systems of which one admits no divergence at all, the definition simplifies to the requirement that the other system may not diverge either.

Definition 13 Let $N = (S, T, F, M_0, \ell)$, $N' = (S', T', F', M'_0, \ell')$ be two nets.

A relation $R \subseteq (\mathbb{N}^S) \times (\mathbb{N}^{S'})$ is a *weak step bisimulation* between N and N' iff

1. $M_0 R M'_0$,
2. $M_1 R M'_1 \wedge M_1 \xrightarrow{A} M_2 \Rightarrow M'_1 \xRightarrow{A} M'_2 \wedge M_2 R M'_2$,
3. $M_1 R M'_1 \wedge M_1 \xrightarrow{\tau} M_2 \Rightarrow M'_1 \xrightarrow{\tau}^* M'_2 \wedge M_2 R M'_2$,
4. $M_1 R M'_1 \wedge M'_1 \xrightarrow{A} M'_2 \Rightarrow M_1 \xRightarrow{A} M_2 \wedge M_2 R M'_2$, and
5. $M_1 R M'_1 \wedge M'_1 \xrightarrow{\tau} M'_2 \Rightarrow M_1 \xrightarrow{\tau}^* M_2 \wedge M_2 R M'_2$,

where $A : \mathbb{N}^{\text{Act}}$.

It is a weak step bisimulation with *explicit divergence* iff furthermore

6. $M_1 R M'_1 \wedge M_1 \xrightarrow{\tau}^\infty \Rightarrow M'_1 \xrightarrow{\tau}^\infty$, and
7. $M_1 R M'_1 \wedge M'_1 \xrightarrow{\tau}^\infty \Rightarrow M_1 \xrightarrow{\tau}^\infty$.

Two nets N and N' are *weakly step bisimilar (with explicit divergence)* iff a weak step bisimulation (with explicit divergence) between them exists. We write $N \approx_{\mathcal{B}} N'$ resp. $N \approx_{\mathcal{B}}^\Delta N'$ iff N and N' are weakly step bisimilar (with explicit divergence).

Restricting the label multisets to singletons in the above definition, one obtains the notions of *weak bisimulation* and *weak bisimilarity* between nets.

Next we define an even finer equivalence than step bisimulation by considering actions not as atomic events but as having a start and an end. An *ST-marking* of a net (S, T, F, M_0, ℓ) is a pair $(M, U) \in \mathbb{N}^S \times T^*$ of a normal marking, together with a sequence of visible transitions *currently firing*. The *initial* ST-marking is $\mathfrak{M}_0 := (M_0, \epsilon)$. The visible behaviour is represented by elements of

$\text{Act}^\pm := \{a^+, a^{-n} \mid a \in \text{Act}, \mathbb{N} \ni n > 0\}$ which are called *visible action phases*, and we define $\text{Act}_\tau^\pm := \text{Act}^\pm \dot{\cup} \{\tau\}$. For $U \in T^*$, we write $t \in^{(n)} U$ if t is the n^{th} element of U . Furthermore U^{-n} denotes U after removal of the n^{th} transition.

Definition 14 Let $N = (S, T, F, M_0, \ell)$ be a net, labelled over Act_τ .

The *ST-transition relations* $\xrightarrow{\eta}$ for $\eta \in \text{Act}_\tau^\pm$ between ST-markings are given by

1. $(M, U) \xrightarrow{a^+} (M', U')$ iff $\exists t \in T. \ell(t) = a \wedge M[t] \wedge M' = M - \bullet t \wedge U' = Ut$.
2. $(M, U) \xrightarrow{a^{-n}} (M', U')$ iff $\exists t \in^{(n)} U. \ell(t) = a \wedge U' = U^{-n} \wedge M' = M + t^\bullet$.
3. $(M, U) \xrightarrow{\tau} (M', U')$ iff $M \xrightarrow{\tau} M' \wedge U' = U$.
4. $(M, U) \xRightarrow{a^+} (M', U')$ iff $(M, U) \xrightarrow{\tau}^* \xrightarrow{a^+} \xrightarrow{\tau}^* (M', U')$.
5. $(M, U) \xRightarrow{a^-} (M', U')$ iff $(M, U) \xrightarrow{\tau}^* \xrightarrow{a^-} \xrightarrow{\tau}^* (M', U')$.

We can now redo the definition of bisimilarity to include the above notions of extended action durations. We also strengthen it to a branching bisimulation.

Definition 15 Let $N = (S, T, F, M_0, \ell)$, $N' = (S', T', F', M'_0, \ell')$ be two nets.

A relation $R \subseteq (\mathbb{N}^S \times T^*) \times (\mathbb{N}^{S'} \times T'^*)$ is a *branching ST-bisimulation* between N and N' iff

1. $(M_0, \varepsilon) R (M'_0, \varepsilon)$,
2. $\mathfrak{M}_1 R \mathfrak{M}'_1 \wedge \mathfrak{M}_1 \xrightarrow{a^+} \mathfrak{M}_2 \Rightarrow \mathfrak{M}'_1 \xrightarrow{\tau}^* \mathfrak{M}'_2 \xrightarrow{a^+} \mathfrak{M}'_3 \wedge \mathfrak{M}_1 R \mathfrak{M}'_2 \wedge \mathfrak{M}_2 R \mathfrak{M}'_3$,
3. $\mathfrak{M}_1 R \mathfrak{M}'_1 \wedge \mathfrak{M}_1 \xrightarrow{a^{-n}} \mathfrak{M}_2 \Rightarrow \mathfrak{M}'_1 \xrightarrow{\tau}^* \mathfrak{M}'_2 \xrightarrow{a^{-n}} \mathfrak{M}'_3 \wedge \mathfrak{M}_1 R \mathfrak{M}'_2 \wedge \mathfrak{M}_2 R \mathfrak{M}'_3$,
4. $\mathfrak{M}_1 R \mathfrak{M}'_1 \wedge \mathfrak{M}_1 \xrightarrow{\tau} \mathfrak{M}_2 \Rightarrow \mathfrak{M}'_1 \xrightarrow{\tau}^* \mathfrak{M}'_2 \xrightarrow{\tau}^? \mathfrak{M}'_3 \wedge \mathfrak{M}_1 R \mathfrak{M}'_2 \wedge \mathfrak{M}_2 R \mathfrak{M}'_3$,
5. $\mathfrak{M}_1 R \mathfrak{M}'_1 \wedge \mathfrak{M}'_1 \xrightarrow{a^+} \mathfrak{M}'_2 \Rightarrow \mathfrak{M}_1 \xrightarrow{\tau}^* \mathfrak{M}_2 \xrightarrow{a^+} \mathfrak{M}_3 \wedge \mathfrak{M}_2 R \mathfrak{M}'_1 \wedge \mathfrak{M}_3 R \mathfrak{M}'_2$,
6. $\mathfrak{M}_1 R \mathfrak{M}'_1 \wedge \mathfrak{M}'_1 \xrightarrow{a^{-n}} \mathfrak{M}'_2 \Rightarrow \mathfrak{M}_1 \xrightarrow{\tau}^* \mathfrak{M}_2 \xrightarrow{a^{-n}} \mathfrak{M}_3 \wedge \mathfrak{M}_2 R \mathfrak{M}'_1 \wedge \mathfrak{M}_3 R \mathfrak{M}'_2$,
7. $\mathfrak{M}_1 R \mathfrak{M}'_1 \wedge \mathfrak{M}'_1 \xrightarrow{\tau} \mathfrak{M}'_2 \Rightarrow \mathfrak{M}_1 \xrightarrow{\tau}^* \mathfrak{M}_2 \xrightarrow{\tau}^? \mathfrak{M}_3 \wedge \mathfrak{M}_2 R \mathfrak{M}'_1 \wedge \mathfrak{M}_3 R \mathfrak{M}'_2$,

where $a \in \text{Act}$ and we write $\mathfrak{M}_i \xrightarrow{\tau}^? \mathfrak{M}_j$ for $\mathfrak{M}_i \xrightarrow{\tau} \mathfrak{M}_j \vee \mathfrak{M}_i = \mathfrak{M}_j$.

A *branching ST-bisimulation with explicit divergence* is defined by additionally requiring

8. if $\mathfrak{M}_1 R \mathfrak{M}'_1$ and there is an infinite sequence of states \mathfrak{M}_i with $\mathbb{N} \ni i \geq 1$ such that $\mathfrak{M}_i \xrightarrow{\tau} \mathfrak{M}_{i+1}$ and $\mathfrak{M}_i R \mathfrak{M}'_1$ for all i , then there exists an infinite sequence of states \mathfrak{M}'_i with $\mathbb{N} \ni i \geq 1$ such that $\mathfrak{M}'_i \xrightarrow{\tau} \mathfrak{M}'_{i+1}$ and $\mathfrak{M}_1 R \mathfrak{M}'_i$ for all i

and

9. if $\mathfrak{M}_1 R \mathfrak{M}'_1$ and there is an infinite sequence of states \mathfrak{M}'_i with $\mathbb{N} \ni i \geq 1$ such that $\mathfrak{M}'_i \xrightarrow{\tau} \mathfrak{M}'_{i+1}$ and $\mathfrak{M}_1 R \mathfrak{M}'_i$ for all i , then there exists an infinite sequence of states \mathfrak{M}_i with $\mathbb{N} \ni i \geq 1$ such that $\mathfrak{M}_i \xrightarrow{\tau} \mathfrak{M}_{i+1}$ and $\mathfrak{M}_i R \mathfrak{M}'_1$ for all i .

Two nets are branching ST-bisimilar (with explicit divergence) iff a branching ST-bisimulation (with explicit divergence) between them exists. We write $N_1 \approx_{bSTb}^\Delta N_2$ iff N_1 and N_2 are branching ST-bisimilar with explicit divergence.

ST-bisimilarity was originally proposed in [GV87]. It was extended to a setting with internal actions in [Vog93].

The next proposition says that branching ST-bisimilarity with explicit divergence is more discriminating than (i. e. *stronger* than, *finer* than, or included in) step failures equivalence.

Proposition 1 Let N_1 and N_2 be nets. If $N_1 \approx_{bSTb}^\Delta N_2$ then $N_1 \approx_{\mathcal{F}} N_2$.

Proof: Suppose $N_1 \approx_{bSTb}^\Delta N_2$ and $\langle \sigma, X \rangle \in \mathcal{F}(N_1)$. By symmetry it suffices to show that $\langle \sigma, X \rangle \in \mathcal{F}(N_2)$.

Since $N_1 \approx_{bSTb}^\Delta N_2$, there must be a branching ST-bisimulation \mathcal{B} between the ST-markings of $N_1 = (S_1, T_1, F_1, M_{01}, \ell_1)$ and $N_2 = (S_2, T_2, F_2, M_{02}, \ell_2)$. In particular, we have $(M_{01}, \epsilon) \mathcal{B} (M_{02}, \epsilon)$. Let $\sigma =: a_1 a_2 \cdots a_n \in \text{Act}^*$. Then $M_{01} \xrightarrow{\tau}^* \xrightarrow{a_1} \xrightarrow{\tau}^* \xrightarrow{a_2} \xrightarrow{\tau}^* \cdots \xrightarrow{\tau}^* \xrightarrow{a_n} \xrightarrow{\tau}^* M'_1$ for a marking $M'_1 \in \mathbb{N}^{S_1}$ with $M'_1 \not\xrightarrow{\tau}$ and $\forall A \in X. M'_1 \not\xrightarrow{A}$. Hence $(M_{01}, \epsilon) \xrightarrow{\tau}^* \xrightarrow{a_1^+} \xrightarrow{a_1^{-1}} \xrightarrow{\tau}^* \xrightarrow{a_2^+} \xrightarrow{a_2^{-1}} \xrightarrow{\tau}^* \cdots \xrightarrow{\tau}^* \xrightarrow{a_n^+} \xrightarrow{a_n^{-1}} \xrightarrow{\tau}^* (M'_1, \epsilon)$. Thus, using the properties of a branching bisimulation between N_1 and N_2 , there must be a marking $M'_2 \in \mathbb{N}^{S_2}$ such that we have $(M_{02}, \epsilon) \xrightarrow{\tau}^* \xrightarrow{a_1^+} \xrightarrow{\tau}^* \xrightarrow{a_1^{-1}} \xrightarrow{\tau}^* \xrightarrow{a_2^+} \xrightarrow{\tau}^* \xrightarrow{a_2^{-1}} \xrightarrow{\tau}^* \cdots \xrightarrow{\tau}^* \xrightarrow{a_n^+} \xrightarrow{\tau}^* \xrightarrow{a_n^{-1}} \xrightarrow{\tau}^* (M'_2, \epsilon)$ and $(M'_1, \epsilon) \mathcal{B} (M'_2, \epsilon)$. Since $(M'_1, \epsilon) \not\xrightarrow{\tau}$, the ST-marking (M'_1, ϵ) admits no divergence. As \approx_{bSTb}^Δ respects this property, also (M'_2, ϵ) admits no divergence, and there must be an $M''_2 \in \mathbb{N}^{S_2}$ with $M''_2 \not\xrightarrow{\tau}$ and $(M'_2, \epsilon) \xrightarrow{\tau}^* (M''_2, \epsilon)$. Clause 7 of Definition 15 then gives us $(M'_1, \epsilon) \mathcal{B} (M''_2, \epsilon)$, and Definition 14 yields $M_{02} \xrightarrow{\sigma} M''_2$.

Now let $B = \{b_1, \dots, b_m\} \in X$. Then $M'_1 \not\xrightarrow{B}$. Suppose, towards a contradiction, that $M''_2 \xrightarrow{B}$. Then $(M''_2, \epsilon) \xrightarrow{b_1^+} \xrightarrow{b_2^+} \cdots \xrightarrow{b_m^+}$. Property 5 of Definition 15 implies $(M'_1, \epsilon) \xrightarrow{b_1^+} \xrightarrow{b_2^+} \cdots \xrightarrow{b_m^+}$ and hence $M'_1 \xrightarrow{B}$. This is a contradiction, so $M''_2 \not\xrightarrow{B}$. It follows that $\langle \sigma, X \rangle \in \mathcal{F}(N_2)$. \square

In this thesis we employ both step failures equivalence and branching ST-bisimilarity with explicit divergence. Fortunately it will turn out that for our purposes the latter equivalence coincides with a simpler variant, branching split bisimilarity (since always one of the compared nets is unlabelled, see Proposition 2).

A *split marking* of a net $N = (S, T, F, M_0, \ell)$ is a pair $(M, U) \in \mathbb{N}^S \times \mathbb{N}^T$ of a normal marking M , together with a multiset of visible transitions currently

firing. The *initial* split marking is $\mathfrak{M}_0 := (M_0, \emptyset)$. A split marking can be regarded as an abstraction from an ST-marking, in which the total order on the (finite) multiset of transitions that are currently firing has been dropped. Let $\text{Act}_{\text{split}}^\pm := \{a^+, a^- \mid a \in \text{Act}\}$.

Definition 16 Let $N = (S, T, F, M_0, \ell)$ be a net, labelled over Act_τ .

The *split transition relations* $\xrightarrow{\zeta}$ for $\zeta \in \text{Act}_{\text{split}}^\pm \dot{\cup} \{\tau\}$ between split markings are given by

1. $(M, U) \xrightarrow{a^+} (M', U')$ iff $\exists t \in T. \ell(t) = a \wedge M[t] \wedge M' = M - \bullet t \wedge U' = U + \{t\}$.
2. $(M, U) \xrightarrow{a^-} (M', U')$ iff $\exists t \in U. \ell(t) = a \wedge U' = U - \{t\} \wedge M' = M + t^\bullet$.
3. $(M, U) \xrightarrow{\tau} (M', U')$ iff $M \xrightarrow{\tau} M' \wedge U' = U$.
4. $(M, U) \xRightarrow{a^+} (M', U')$ iff $(M, U) \xrightarrow{\tau}^* \xrightarrow{a^+} \xrightarrow{\tau}^* (M', U')$.
5. $(M, U) \xRightarrow{a^-} (M', U')$ iff $(M, U) \xrightarrow{\tau}^* \xrightarrow{a^-} \xrightarrow{\tau}^* (M', U')$.

Note that $(M, U) \xrightarrow{a^+}$ iff $M \xrightarrow{a}$, whereas $(M, U) \xrightarrow{a^-}$ iff $a \in \ell(U)$. With induction on reachability of markings it is furthermore easy to check that $(M, U) \in [\mathfrak{M}_0]$ iff $\tau \notin \ell(U)$ and $M + \bullet U \in [M_0]$.

Definition 17 Redoing Definition 15 with the transition relation given in Definition 16 (and substituting \xrightarrow{a} for \xRightarrow{a}), we obtain *branching split bisimilarity* (with explicit divergence).

For $\mathfrak{M} = (M, U) \in \mathbb{N}^S \times T^*$ an ST-marking, let $\overline{\mathfrak{M}} = (M, \overline{U}) \in \mathbb{N}^S \times \mathbb{N}^T$ be the split marking obtained by converting the sequence U into the multiset \overline{U} , where $\overline{U}(t)$ is the number of occurrences of the transition $t \in T$ in U . Moreover, define $\ell(\mathfrak{M})$ by $\ell(M, U) := \ell(U)$ and $\ell(t_1 t_2 \dots t_k) := \ell(t_1) \ell(t_2) \dots \ell(t_k)$. Furthermore, for $\eta \in \text{Act}_\tau^\pm$, let $\overline{\eta} \in \text{Act}_{\text{split}}^\pm \dot{\cup} \{\tau\}$ be given by $\overline{a^+} := a^+$, $\overline{a^-} := a^-$ and $\overline{\tau} := \tau$. We also write $\mathfrak{M} \xrightarrow{(\alpha)} \mathfrak{M}'$ to denote $\mathfrak{M} \xrightarrow{\alpha} \mathfrak{M}' \vee (\alpha = \tau \wedge \mathfrak{M} = \mathfrak{M}')$, meaning that in case $\alpha = \tau$ performing a τ -transition is optional.

Observation 1 Let $\mathfrak{M}, \mathfrak{M}'$ be ST-markings, \mathfrak{M}^\dagger a split marking, $\eta \in \text{Act}_\tau^\pm$ and $\zeta \in \text{Act}_{\text{split}}^\pm \dot{\cup} \{\tau\}$. Then

1. $\mathfrak{M} \in \mathbb{N}^S \times T^*$ is the initial ST-marking of N iff $\overline{\mathfrak{M}} \in \mathbb{N}^S \times \mathbb{N}^T$ is the initial split marking of N ;
2. if $\mathfrak{M} \xrightarrow{\eta} \mathfrak{M}'$ then $\overline{\mathfrak{M}} \xrightarrow{\overline{\eta}} \overline{\mathfrak{M}'}$;
3. if $\overline{\mathfrak{M}} \xrightarrow{\zeta} \mathfrak{M}^\dagger$ then there is a $\mathfrak{M}' \in \mathbb{N}^S \times T^*$ and $\eta \in \text{Act}_\tau^\pm$ such that $\mathfrak{M} \xrightarrow{\eta} \mathfrak{M}'$, $\overline{\eta} = \zeta$ and $\overline{\mathfrak{M}'} = \mathfrak{M}^\dagger$;
4. if $\mathfrak{M} \xrightarrow{(\eta)} \mathfrak{M}'$ then $\overline{\mathfrak{M}} \xrightarrow{(\overline{\eta})} \overline{\mathfrak{M}'}$;
5. if $\overline{\mathfrak{M}} \xrightarrow{(\zeta)} \mathfrak{M}^\dagger$ then there is a $\mathfrak{M}' \in \mathbb{N}^S \times T^*$ and $\eta \in \text{Act}_\tau^\pm$ such that $\mathfrak{M} \xrightarrow{(\eta)} \mathfrak{M}'$, $\overline{\eta} = \zeta$ and $\overline{\mathfrak{M}'} = \mathfrak{M}^\dagger$;

6. if $\mathfrak{M} \xrightarrow{\tau}^* \mathfrak{M}'$ then $\overline{\mathfrak{M}} \xrightarrow{\tau}^* \overline{\mathfrak{M}'}$;
7. if $\overline{\mathfrak{M}} \xrightarrow{\tau}^* \mathfrak{M}^\dagger$ then there is a $\mathfrak{M}' \in \mathbb{N}^S \times T^*$ such that $\mathfrak{M} \xrightarrow{\tau}^* \mathfrak{M}'$ and $\overline{\mathfrak{M}'} = \mathfrak{M}^\dagger$. \square

Lemma 2 Let $N_1 = (S_1, T_1, F_1, M_{01}, \ell_1)$ and $N_2 = (S_2, T_2, F_2, M_{02}, \ell_2)$ be two nets, N_2 being unlabelled; let \mathfrak{M}_1 and \mathfrak{M}'_1 be ST-markings of N_1 , and $\mathfrak{M}_2, \mathfrak{M}'_2$ ST-markings of N_2 . If $\ell_2(\mathfrak{M}_2) = \ell_1(\mathfrak{M}_1)$, $\mathfrak{M}_1 \xrightarrow{\eta} \mathfrak{M}'_1$ and $\mathfrak{M}_2 \xrightarrow{\eta'} \mathfrak{M}'_2$ with $\overline{\eta'} = \overline{\eta}$, then there is an \mathfrak{M}''_2 with $\mathfrak{M}_2 \xrightarrow{\eta} \mathfrak{M}''_2$, $\ell_2(\mathfrak{M}''_2) = \ell_1(\mathfrak{M}'_1)$, and $\overline{\mathfrak{M}''_2} = \overline{\mathfrak{M}'_2}$.

Proof: If $\mathfrak{M} \xrightarrow{\eta} \mathfrak{M}'$ or $\mathfrak{M} \xrightarrow{(\eta)} \mathfrak{M}'$ then $\ell_i(\mathfrak{M}')$ is completely determined by $\ell_i(\mathfrak{M})$ and η . For this reason the requirement $\ell_2(\mathfrak{M}''_2) = \ell_1(\mathfrak{M}'_1)$ will hold as soon as the other requirements are met.

First suppose η is of the form τ or a^+ . Then $\overline{\eta} = \eta$ and moreover $\overline{\eta'} = \overline{\eta}$ implies $\eta' = \eta$. Thus we can take $\mathfrak{M}''_2 := \mathfrak{M}'_2$.

Now suppose $\eta := a^{-n}$ for some $n > 0$. Then $\eta' = a^{-m}$ for some $m > 0$. As $\mathfrak{M}_1 \xrightarrow{\eta}$, the n^{th} element of $\ell_1(\mathfrak{M}_1)$ must (exist and) be a . Since $\ell_2(\mathfrak{M}_2) = \ell_1(\mathfrak{M}_1)$, also the n^{th} element of $\ell_2(\mathfrak{M}_2)$ must be a , so there is an \mathfrak{M}''_2 with $\mathfrak{M}_2 \xrightarrow{\eta} \mathfrak{M}''_2$. Let $\mathfrak{M}_2 := (M_2, U_2)$. Then U_2 is a sequence of transitions of which the n^{th} and the m^{th} elements are both labelled a . Since the net N_2 is unlabelled, those two transitions must be equal. Let $\mathfrak{M}'_2 := (M'_2, U'_2)$ and $\mathfrak{M}''_2 := (M''_2, U''_2)$. We find that $M'_2 = M''_2$ and $\overline{U'_2} = \overline{U''_2}$. It follows that $\overline{\mathfrak{M}''_2} = \overline{\mathfrak{M}'_2}$. \square

Observation 2 If $\mathfrak{M} \xrightarrow{\tau}^* \mathfrak{M}'$ for ST-markings $\mathfrak{M}, \mathfrak{M}'$ then $\ell(\mathfrak{M}') = \ell(\mathfrak{M})$. \square

Observation 3 If $\ell(\mathfrak{M}_1) = \ell(\mathfrak{M}_2)$ and $\mathfrak{M}_2 \xrightarrow{a^{-n}}$ for some $a \in \text{Act}$ and $n > 0$, then $\mathfrak{M}_1 \xrightarrow{a^{-n}}$. \square

Observation 4 If $\mathfrak{M} \xrightarrow{a^{-n}} \mathfrak{M}'$ and $\mathfrak{M} \xrightarrow{a^{-n}} \mathfrak{M}''$ for some $a \in \text{Act}$ and $n > 0$, then $\mathfrak{M}' = \mathfrak{M}''$. \square

Proposition 2 Let $N_1 = (S_1, T_1, F_1, M_{01}, \ell_1)$ and $N_2 = (S_2, T_2, F_2, M_{02}, \ell_2)$ be two nets, N_2 being unlabelled. Then N_1 and N_2 are branching ST-bisimilar (with explicit divergence) iff they are branching split bisimilar (with explicit divergence).

Proof: Suppose \mathcal{B} is a branching ST-bisimulation between N_1 and N_2 . Then, by Observation 1, the relation $\mathcal{B}_{\text{split}} := \{(\overline{\mathfrak{M}_1}, \overline{\mathfrak{M}_2}) \mid (\mathfrak{M}_1, \mathfrak{M}_2) \in \mathcal{B}\}$ is a branching split bisimulation between N_1 and N_2 .

Now let \mathcal{B} be a branching split bisimulation between N_1 and N_2 . Then, using Observation 1, $\mathcal{B}_{\text{ST}} := \{(\mathfrak{M}_1, \mathfrak{M}_2) \mid \ell_1(\mathfrak{M}_1) = \ell_2(\mathfrak{M}_2) \wedge (\overline{\mathfrak{M}_1}, \overline{\mathfrak{M}_2}) \in \mathcal{B}\}$ turns out to be a branching ST-bisimulation between N_1 and N_2 :

1. $\mathfrak{M}_{o1} \mathcal{B}_{\text{ST}} \mathfrak{M}_{o2}$ follows from Observation 1(1), since $\overline{\mathfrak{M}_{o1}} \mathcal{B} \overline{\mathfrak{M}_{o2}}$ and $\ell_1(\mathfrak{M}_{o1}) = \ell_2(\mathfrak{M}_{o2}) = \epsilon$.

2. Suppose $\mathfrak{M}_1 \mathcal{B}_{\text{ST}} \mathfrak{M}_2$ and $\mathfrak{M}_1 \xrightarrow{\eta} \mathfrak{M}'_1$. Then $\overline{\mathfrak{M}_1} \mathcal{B} \overline{\mathfrak{M}_2}$ and $\overline{\mathfrak{M}_1} \xrightarrow{\overline{\eta}} \overline{\mathfrak{M}'_1}$. Hence $\exists \mathfrak{M}_2^\dagger, \mathfrak{M}_2^\ddagger$ such that $\overline{\mathfrak{M}_2} \xrightarrow{\tau}^* \mathfrak{M}_2^\dagger \xrightarrow{(\overline{\eta})} \mathfrak{M}_2^\ddagger$, $\overline{\mathfrak{M}_1} \mathcal{B} \mathfrak{M}_2^\dagger$ and $\overline{\mathfrak{M}'_1} \mathcal{B} \mathfrak{M}_2^\ddagger$. As N_2 is unlabelled, $\mathfrak{M}_2^\dagger = \overline{\mathfrak{M}_2}$. By Observation 1(5), using that $\overline{\mathfrak{M}_2} \xrightarrow{(\overline{\eta})} \mathfrak{M}_2^\ddagger$, there exist \mathfrak{M}'_2, η' such that $\mathfrak{M}_2 \xrightarrow{(\eta')} \mathfrak{M}'_2$, $\overline{\eta'} = \overline{\eta}$ and $\overline{\mathfrak{M}'_2} = \mathfrak{M}_2^\ddagger$. By Lemma 2, there is an ST-marking \mathfrak{M}_2'' such that $\mathfrak{M}_2 \xrightarrow{(\eta)} \mathfrak{M}_2''$, $\ell_2(\mathfrak{M}_2'') = \ell_1(\mathfrak{M}'_1)$, and $\overline{\mathfrak{M}_2''} = \overline{\mathfrak{M}_2} = \mathfrak{M}_2^\ddagger$. It follows that $\mathfrak{M}'_1 \mathcal{B}_{\text{ST}} \mathfrak{M}_2''$.
3. Suppose $\mathfrak{M}_1 \mathcal{B}_{\text{ST}} \mathfrak{M}_2$ and $\mathfrak{M}_2 \xrightarrow{\eta} \mathfrak{M}'_2$. Then $\overline{\mathfrak{M}_1} \mathcal{B} \overline{\mathfrak{M}_2}$ and $\overline{\mathfrak{M}_2} \xrightarrow{\overline{\eta}} \overline{\mathfrak{M}'_2}$. Hence $\exists \mathfrak{M}_1^\dagger, \mathfrak{M}_1^\ddagger$ such that $\overline{\mathfrak{M}_1} \xrightarrow{\tau}^* \mathfrak{M}_1^\dagger \xrightarrow{(\overline{\eta})} \mathfrak{M}_1^\ddagger$, $\mathfrak{M}_1^\dagger \mathcal{B} \overline{\mathfrak{M}_2}$ and $\mathfrak{M}_1^\ddagger \mathcal{B} \overline{\mathfrak{M}'_2}$. By Observation 1(7), $\exists \mathfrak{M}_1^*$ such that $\mathfrak{M}_1 \xrightarrow{\tau}^* \mathfrak{M}_1^*$ and $\overline{\mathfrak{M}_1^*} = \mathfrak{M}_1^\ddagger$. By Observation 2, $\ell_1(\mathfrak{M}_1^*) = \ell_1(\mathfrak{M}_1) = \ell_2(\mathfrak{M}_2)$, so $\mathfrak{M}_1^* \mathcal{B}_{\text{ST}} \mathfrak{M}_2$. Since N_2 is unlabelled, $\eta \neq \tau$.
 - Let $\eta = a^+$ for some $a \in \text{Act}$. Using that $\overline{\mathfrak{M}_1^*} \xrightarrow{(\overline{\eta})} \mathfrak{M}_1^\ddagger$, by Observation 1(5) $\exists \mathfrak{M}'_1, \eta'$ such that $\mathfrak{M}_1^* \xrightarrow{(\eta')} \mathfrak{M}'_1$, $\overline{\eta'} = \overline{\eta}$ and $\overline{\mathfrak{M}'_1} = \mathfrak{M}_1^\ddagger$. It must be that $\eta' = \eta = a^+$ and $\ell_1(\mathfrak{M}'_1) = \ell_1(\mathfrak{M}_1^*)a = \ell_2(\mathfrak{M}_2)a = \ell_2(\mathfrak{M}'_2)$. Hence $\mathfrak{M}'_1 \mathcal{B}_{\text{ST}} \mathfrak{M}'_2$.
 - Let $\eta = a^{-n}$ for some $a \in \text{Act}$ and $n > 0$. By Observation 3, $\exists \mathfrak{M}'_1$ with $\mathfrak{M}_1^* \xrightarrow{\eta} \mathfrak{M}'_1$. By Part 2. of this proof, $\exists \mathfrak{M}_2''$ such that $\mathfrak{M}_2 \xrightarrow{(\eta)} \mathfrak{M}_2''$ and $\mathfrak{M}'_1 \mathcal{B}_{\text{ST}} \mathfrak{M}_2''$. By Observation 4 $\mathfrak{M}_2'' = \mathfrak{M}'_2$.

Since the net N_2 is unlabelled, it has no divergence. In such a case, the requirement “with explicit divergence” requires N_1 to be free of divergence as well, regardless of whether split or ST-semantics is used. \square

Bisimulation based equivalences provide very fine discrimination between systems of different branching structure. When focusing mainly instead on the causal structure of a system, i.e. how actions are depending on each other, we employ another equivalence: Completed pomset trace equivalence, an extension of the pomset trace equivalence of [Pra85], disregards the branching structure of a system except for local and global deadlocks, but respects the causal relationships between all actions.

Completed pomset trace equivalence is obtained by unrolling a net into a process as defined in [Pet77]. Such a process can be understood to be an account of one particular way to decide all conflicts which occurred while proceeding from one marking to the next. The behaviour of the net is hence a set of these processes, covering all possible ways to decide conflicts.

Unrolling a net N intuitively proceeds as follows: The initially marked places of N are copied into a new net \mathcal{N} and their correspondence to the original places recorded in a mapping π . Then, whenever in N a transition t is fired, this is replayed in \mathcal{N} by a new transition connected to places corresponding by π to the original preplaces of t and which are not yet connected to any other posttransition. A new place of \mathcal{N} is created for every token produced by t . Again all correspondences are recorded in π . Every place of \mathcal{N} has thus at most one posttransition. If it has none, this place represents a token currently residing on the corresponding original place.

As a shorthand notation to gather these places, we introduce the *end* of a net.

Definition 18 Let $N = (S, T, F, M_0, \ell)$ be a labelled net. The *end* of the net is defined as $N^\circ := \{s \in S \mid s^\bullet = \emptyset\}$.

Definition 19

A pair $\mathcal{P} = (\mathcal{N}, \pi)$ is a *process* of a net $N = (S, T, F, M_0, \ell)$ iff

- $\mathcal{N} = (\mathcal{P}, \mathcal{T}, \mathcal{F}, \mathcal{M}_0, \ell)$ is a net, satisfying
 - $\forall s \in \mathcal{P}. |\bullet s| \leq 1 \geq |s^\bullet| \wedge \mathcal{M}_0(s) = \begin{cases} 1 & \text{iff } \bullet s = \emptyset \\ 0 & \text{otherwise} \end{cases}$,
 - all arc-weights are 1, i.e. $\mathcal{F}(x, y) \in \{0, 1\}$ for all x, y and \mathcal{F} can be considered a relation,
 - \mathcal{F} is acyclic, i.e. $\forall x \in \mathcal{P} \cup \mathcal{T}. (x, x) \notin \mathcal{F}^+$, where \mathcal{F}^+ is the transitive closure of \mathcal{F} ,
 - and $\{t \mid (t, u) \in \mathcal{F}^+\}$ is finite for all $u \in \mathcal{T}$.
- $\pi : \mathcal{P} \cup \mathcal{T} \rightarrow S \cup T$ is a function with $\pi(\mathcal{P}) \subseteq S$ and $\pi(\mathcal{T}) \subseteq T$, satisfying
 - $|\pi^{-1}(s) \cap \mathcal{M}_0| = M_0(s)$ for all $s \in S$,
 - $\forall t \in \mathcal{T}, s \in S. F(s, \pi(t)) = |\pi^{-1}(s) \cap \bullet t| \wedge F(\pi(t), s) = |\pi^{-1}(s) \cap t^\bullet|$,
and
 - $\forall t \in \mathcal{T}. \ell(t) = \ell(\pi(t))$.³

\mathcal{P} is called *finite* if \mathcal{N} is finite. \mathcal{P} is *maximal* iff $\nexists G. \pi(\mathcal{N}^\circ)[G]_N$. The set of all maximal processes of a net N is denoted by $MP(N)$.

To disambiguate between a not-yet-occurred firing of a transition a and the impossibility of firing an a , we restrict the set of processes relevant for the behavioural description to maximal processes. We thereby obtain a just semantics in the sense of Reisig [Rei84]⁴, i.e. a transition which remained enabled infinitely long must ultimately fire.

The conditions for \mathcal{N} ensure that a process is indeed a mapping from an occurrence net as defined in [Pet77, GSW80] to the net N ; hence we have defined processes here in the classical way as in [GR83, BD87] (even though not introducing occurrence nets explicitly).

To abstract from the τ -actions introduced in an implementation, we extract from the maximal processes the causal structure between the fired visible events in the form of a partially ordered multiset (*pomset*). Formally, a pomset is an isomorphism class of a partially ordered multiset of action labels.

³While ℓ and ℓ look nearly identical, the authors see no problem in that, given the close correspondence.

⁴or in modern terms, a “weakly fair” semantics

Definition 20 A *labelled partial order* is a structure (V, T, \leq, l) where

- V is a set (of *vertices*),
- T is a set (of *labels*),
- $\leq \subseteq V \times V$ is a partial order relation and
- $l : V \rightarrow T$ (the *labelling function*).

Two labelled partial orders $o = (V, T, \leq, l)$ and $o' = (V', T, \leq', l')$ are *isomorphic*, $o \cong o'$, iff there exists a bijection $\varphi : V \rightarrow V'$ such that

- $\forall v \in V. l(v) = l'(\varphi(v))$ and
- $\forall u, v \in V. u \leq v \Leftrightarrow \varphi(u) \leq' \varphi(v)$.

Definition 21 Let $o = (V, T, \leq, l)$ be a partial order. The *pomset* of o is its isomorphism class $[o] := \{o' \mid o \cong o'\}$.

By hiding the unobservable transitions of a process, we gain a pomset which describes causality relations of all participating visible transitions.

Definition 22 Let $\mathcal{P} = ((\mathcal{S}, \mathcal{T}, \mathcal{F}, \mathcal{M}_0, \ell), \pi)$ be a process. Let $\mathcal{O} := \{t \in \mathcal{T} \mid \ell(t) \neq \tau\}$, i. e. the visible transitions of the process. The *visible pomset* of \mathcal{P} is the pomset $VP(\mathcal{P}) := [(\mathcal{O}, \text{Act}, \mathcal{F}^* \cap \mathcal{O} \times \mathcal{O}, \ell \cap (\mathcal{O} \times \text{Act}))]$ where \mathcal{F}^* is the transitive and reflexive closure of the flow relation \mathcal{F} .

$\text{MVP}(N) := \{VP(\mathcal{P}) \mid \mathcal{P} \in MP(N)\}$ is the set of visible pomsets of all maximal processes of N .

Using this notion we can now define completed pomset trace equivalence.

Definition 23 Two nets N and N' are *completed pomset trace equivalent*, $N \approx_{cPT} N'$, iff $\text{MVP}(N) = \text{MVP}(N')$.

In Section 6.2, we want to track causality throughout the evolution of a net, but restrict attention to 1-safe nets. To this end we extend the usual notion of marking to *dependency marking*. Within these dependency markings, every token is augmented with the labels of all transitions having causally contributed to its existence. In a way, this can be seen as a very coarse abstraction over pomset traces, which – crucially – results in a finite state space for finite 1-safe nets. The basic Petri net notions presented above can then be extended in the same manner.

Definition 24 Let $N = (S, T, F, M_0, \ell)$ be a net. Let $M_1, M_2 \subseteq S \times 2^{\text{Act}}$ and let pr_1, pr_2 be the projection functions to the first and second component, respectively. $G \subseteq T, G \neq \emptyset$, is called a *dependency step* from M_1 to M_2 , $M_1[G]_N^d M_2$, iff

- G is a step from M_1 to M_2 when disregarding the causalities, i. e. $\text{pr}_1(M_1)[G]_N \text{pr}_1(M_2)$, and

- causalities are extended by the labels of the firing transitions:

$$M_2 = \{p \in M_1 \mid \text{pr}_1(p) \notin \bullet G\} \cup \left\{ \left(s, (\{\ell(t)\} \setminus \{\tau\}) \cup \bigcup_{\text{pr}_1(p) \in \bullet t} \text{pr}_2(p) \right) \mid t \in G, s \in t^\bullet \right\}.$$

Applying pr_1 to a dependency marking results in the classical Petri net notion of marking. Note that the enrichment of markings into dependency markings has no impact on the existence of steps, since it neither influences the enabling of transitions nor their independence. A dependency token $(s, P) \in M$ is Q -dependent iff $Q \subseteq P$ and Q -independent iff $P \cap Q = \emptyset$. Where appropriate, we will put a superscript d on symbols to denote their obvious extension to dependency markings, giving us in particular $[M_0]^d$ and \xRightarrow{A}^d .

Chapter 3

Distributed Nets

The following chapter consists mainly of the work published in [GGS08a].

In nets, an inherent concept of simultaneity is built in, since when a transition has more than one preplace, it can be crucial that tokens are removed instantaneously. When using a net to model a system which is intended to be implemented in a distributed way, this built-in concept of synchronous interaction may be problematic.

In this chapter, a given net is regarded as a *specification* of how a system should behave, and this specification involves complete synchronisation of the firing of a transition and the removal of all tokens from its preplaces. We propose various definitions of an *asynchronous implementation* of a net N , in which such synchronous interaction is wholly or partially ruled out and replaced by asynchronous interaction. The question to be clarified is whether such an asynchronous implementation faithfully mimics the dynamic behaviour of N . If this is the case, we call the net N *asynchronous* with respect to the chosen interaction pattern.

The resulting concept of asynchrony is parametrised by the answers to three questions:

1. Which synchronous interactions do we want to rule out exactly?
2. How do we replace synchronous by asynchronous interaction?
3. When does one net faithfully mimic the dynamic behaviour of another?

3.1 Asynchronous Net Classes

To answer the first question we associate a *location* to each place and each transition in a net. A transition may take a token instantaneously from a preplace (when firing) iff this preplace is co-located with the transition; if the preplace resides on a different location than the transition, we have to assume the collection of the token takes time, and thus the place loses its token *before* the transition fires.

We model the association of locations to the places and transitions in a net $N = (S, T, F, M_0, \ell)$ as a function $D : S \cup T \rightarrow \text{Loc}$, with Loc a set of possible locations. We refer to such a function as a *distribution* of N . Since the identity of the locations is irrelevant for our purposes, we can just as well abstract from Loc and represent D by the equivalence relation \equiv_D on $S \cup T$ given by $x \equiv_D y$ iff $D(x) = D(y)$.

In this thesis we do not deal with nets that have a distribution built in. We characterise the interaction patterns we are interested in by imposing particular restrictions on the allowed distributions. The implementor of a net can choose any distribution that satisfies the chosen requirements, and we call a net asynchronous for a certain interaction pattern if it has a correct asynchronous implementation based on any distribution satisfying the respective requirements.

The *fully asynchronous* interaction pattern is obtained by requiring that all places and all transitions reside on different locations. This makes it necessary to implement the removal of every token in a time-consuming way. However, this leads to a rather small class of asynchronous nets, that falls short for many applications. We therefore propose two ways to loosen this requirement, thereby building a hierarchy of classes of asynchronous nets. Both require that all places reside on different locations, but a transition may be co-located with one of its preplaces. The *symmetrically asynchronous* interaction pattern allows this only for transitions with a single preplace, whereas in the *asymmetrically asynchronous* interaction pattern any transition may be co-located with one of its preplaces. Since two preplaces can never be co-located, this breaks the symmetry between the preplaces of a transition; an implementor of a net has to choose at most one preplace for every transition, and co-locate the transition with it. The removal of tokens from all other preplaces needs to be implemented in a time-consuming way. Note that all three interaction patterns break the synchronisation of the token removal between the various preplaces.

Definition 25 Let D be a distribution on a net $N = (S, T, F, M_0, \ell)$, and let \equiv_D be the induced equivalence relation on $S \cup T$. We say that D is

- *fully distributed*, $D \in \mathcal{Q}_{\text{FD}}$, when $x \equiv_D y$ for $x, y \in S \cup T$ only if $x = y$,
- *symmetrically distributed*, $D \in \mathcal{Q}_{\text{SD}}$, when

$$\begin{array}{lll} p \equiv_D q & \text{for } p, q \in S & \text{only if } p = q, \\ t \equiv_D p & \text{for } t \in T, p \in S & \text{only if } \bullet t = \{p\} \text{ and} \\ t \equiv_D u & \text{for } t, u \in T & \text{only if } t = u \text{ or } \exists p \in S. t \equiv_D p \equiv_D u, \end{array}$$

- *asymmetrically distributed*, $D \in \mathcal{Q}_{\text{AD}}$, when

$$\begin{array}{lll} p \equiv_D q & \text{for } p, q \in S & \text{only if } p = q, \\ t \equiv_D p & \text{for } t \in T, p \in S & \text{only if } p \in \bullet t \text{ and} \\ t \equiv_D u & \text{for } t, u \in T & \text{only if } t = u \text{ or } \exists p \in S. t \equiv_D p \equiv_D u. \end{array}$$

The second question raised above was: How do we replace synchronous by asynchronous interaction? In this section we assume that if an arc goes from a

place s to a transition t at a different location, a token takes time to move from s to t . Formally, we describe this by inserting silent (unobservable) transitions between transitions and their remote preplaces. This leads to the following notion of an asynchronous implementation of a net with respect to a chosen distribution.

Definition 26 Let $N = (S, T, F, M_0, \ell)$ be a net, and let \equiv_D be an equivalence relation on $S \cup T$.

The D -based asynchronous implementation of N is $I_D(N) := (S \cup S^\tau, T \cup T^\tau, F', M_0, \ell')$ with

$$\begin{aligned} S^\tau &:= \{s_t \mid t \in T, s \in \bullet t, s \not\equiv_D t\}, \\ T^\tau &:= \{t_s \mid t \in T, s \in \bullet t, s \not\equiv_D t\}, \\ F'(x, y) &:= \begin{cases} F(x, y) & \text{if } x \in T \wedge y \in S \\ F(x, y) & \text{if } x \in S \wedge y \in T \wedge s \equiv_D t \\ F(s, t) & \text{if } x = s \in S \wedge y = t_s \in T^\tau \\ 1 & \text{if } x = t_s \in T^\tau \wedge y = s_t \in S^\tau \\ 1 & \text{if } x = s_t \in S^\tau \wedge y = t \in T \\ 0 & \text{otherwise,} \end{cases} \\ &\text{and} \\ \ell'(t) &:= \begin{cases} \ell(t) & \text{if } t \in T \\ \tau & \text{otherwise.} \end{cases} \end{aligned}$$

For better readability we will use the abbreviations $^\circ x$ and x° defined as $(^\circ x)(y) := F'(y, x)$ and $(x^\circ)(y) := F'(x, y)$ instead of $\bullet x$ or x^\bullet when making assertions about the flow relation of an implementation.

Proposition 3 Let $N = (S, T, F, M_0, \ell)$ be a net and \equiv_D an equivalence relation on $S \cup T$.

1. $I_D(N)$ is a net,
2. $I_D(N)$ is finite iff N is,
3. $I_D(N)$ is finitary iff N is,
4. $I_D(N)$ is plain labelled iff N is.

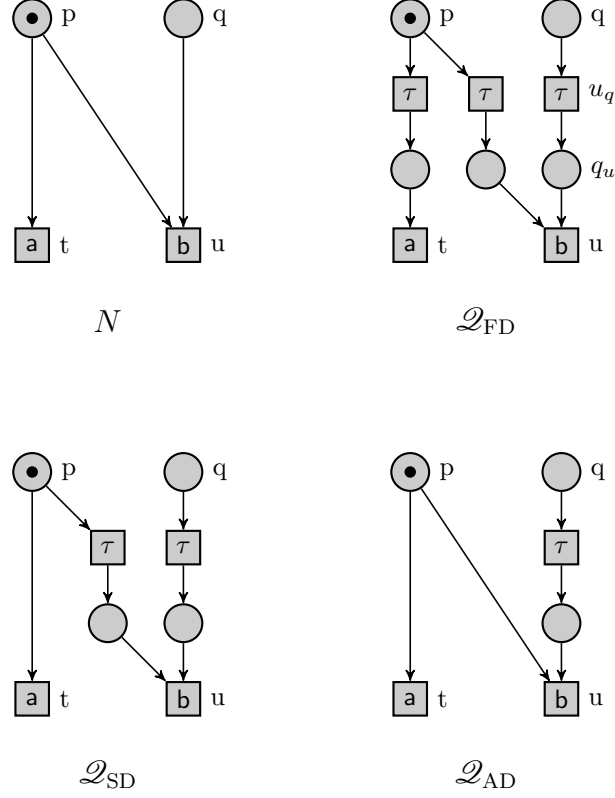
Proof: We use S^τ, T^τ, F' , and ℓ' as in Definition 26.

(1): From $S \cap T = \emptyset$ follows $S^\tau \cap (T \cup T^\tau) = \emptyset$ and $T^\tau \cap (S \cup S^\tau) = \emptyset$. All other requirements follow even more immediately.

(2): If S and T are both finite, so are S^τ and T^τ .

(3): $t^\circ = t^\bullet$ for all $t \in T$ and $|t_s^\circ| = 1$ for all $t_s \in T^\tau$. $|s^\circ| = |s^\bullet|$ for all $s \in S$ and $|s_t^\circ| = 1$ for all $s_t \in S^\tau$. Hence x° is finite for all $x \in (S \cup S^\tau \cup T \cup T^\tau)$. Finally, $^\circ t_s = \{s_t\}$ for all $t_s \in T^\tau$ and if $s \in \bullet t$ in N then either $s \in {}^\circ t$ or $s_t \in {}^\circ t$. Hence $^\circ t \neq \emptyset$ for all $t \in T \cup T^\tau$.

(4): $\ell'(T^\tau) \cap \text{Act} = \emptyset$, i. e. no new labelled transitions were introduced. \square

Figure 3.1: Possible results for $I_D(N)$ given different requirements

The above protocol for replacing synchronous by asynchronous interaction appears to be one of the simplest ones imaginable. More intricate protocols, involving many asynchronous messages between a transition and its preplaces, could be contemplated, but we will not study them yet. Our protocol involves just one such message, namely from the preplace to its posttransition. The protocol is illustrated in Figure 3.1.

The last question above was: When does one net faithfully mimic the dynamic behaviour of another? This asks for a *semantic equivalence* on nets, telling when two nets display the same behaviour. Many such equivalences have been studied in the literature. We believe that most of our results are independent of the precise choice of a semantic equivalence, as long as it preserves causality and branching time to some degree, and abstracts from silent transitions.¹ Therefore we choose one such equivalence, based on its technical convenience in establishing our results, and postpone questions on the effect of varying this equivalence for further research. Our choice is *step readiness*

¹For results on linear-time equivalences, see Section 6.2 and my earlier work in [Sch08, Sch09].

equivalence, as defined in Section 2.2. Using this equivalence, we define a notion of *behavioural asynchrony* by asking whether the asynchronous implementation of a net preserves its behaviour. This notion is parametrised by the chosen interaction pattern, characterised as a requirement on the allowed distributions.

Definition 27 Let \mathcal{Q} be a requirement on distributions of nets.

An unlabelled net N is *behaviourally \mathcal{Q} -asynchronous* iff there exists a distribution D of N meeting the requirement \mathcal{Q} such that $I_D(N) \approx_{\mathcal{R}} N$.

Intuitively, the only behavioural difference between a net N and its asynchronous implementation $I_D(N)$ can occur when in N a place $s \in \bullet u$ is marked, whereas in $I_D(N)$ this token is already on its way from s to its posttransition u . In that case, it may occur that a transition $t \neq u$ with $s \in \bullet t$ is enabled in N , whereas t is not enabled in the described state of $I_D(N)$. We call the situation in N leading to this state of $I_D(N)$ a *distributed conflict*; it is in fact the only circumstance in which $I_D(N)$ fails to faithfully mimic the dynamic behaviour of N .

Definition 28 Let $N = (S, T, F, M_0, \ell)$ be a net and D a distribution of N .

N has a *distributed conflict with respect to D* iff

$$\begin{aligned} \exists t, u \in T \exists p \in \bullet t \cap \bullet u. t \neq u \wedge p \not\equiv_D u \wedge \exists M \in [M_0]_N \exists k, l \in \mathbb{N}. \\ k \cdot \bullet t \leq M \wedge l \cdot \bullet u(p) \leq M(p) \wedge k \cdot \bullet t(p) \not\leq M(p) - l \cdot \bullet u(p). \end{aligned}$$

Observation 5 If $N = (S, T, F, M_0, \ell)$ is a 1-safe net without arc weights and D a distribution thereof, then N has a distributed conflict with respect to D iff

$$\exists t, u \in T \exists p \in \bullet t \cap \bullet u. t \neq u \wedge p \not\equiv_D u \wedge \exists M \in [M_0]_N. \bullet t \leq M. \quad \square$$

We wish to call a net N *(semi)structurally asynchronous* iff the situation outlined above never occurs, so that the asynchronous implementation does not change the behaviour of the net. As for behavioural asynchrony, this notion of asynchrony is parametrised by the set of allowed distributions.

Definition 29 Let \mathcal{Q} be a requirement on distributions of nets.

A net N is *(semi)structurally \mathcal{Q} -asynchronous* iff there exists a distribution D of N meeting the requirement \mathcal{Q} such that N has no distributed conflicts with respect to D .

The following theorem shows that distributed conflicts describe exactly the critical situations: For all unlabelled nets the notions of structural and behavioural asynchrony coincide, regardless of the choice of \mathcal{Q} .

Theorem 1 Let N be an unlabelled, finitary net, and \mathcal{Q} a requirement on distributions of nets.

N is behaviourally \mathcal{Q} -asynchronous iff it is structurally \mathcal{Q} -asynchronous.

Proof: In Section 3.2. \square

Because of this theorem, we call an unlabelled net \mathcal{Q} -asynchronous if it is behaviourally and/or structurally \mathcal{Q} -asynchronous. In this thesis we study this concept for unlabelled nets only. When taking $\mathcal{Q} = \mathcal{Q}_{\text{FD}}$ we speak of *fully asynchronous nets*, when taking $\mathcal{Q} = \mathcal{Q}_{\text{SD}}$ of *symmetrically asynchronous nets*, and when taking $\mathcal{Q} = \mathcal{Q}_{\text{AD}}$ of *asymmetrically asynchronous nets*.

Example 1 The net N of Figure 3.1 is not fully asynchronous, for its unique D -based asynchronous implementation $I_D(N)$ with $D \in \mathcal{Q}_{\text{FD}}$ (also displayed in Figure 3.1) is not step readiness equivalent to N . In fact $\langle \varepsilon, \emptyset \rangle \in \mathcal{R}(I_D(N)) \setminus \mathcal{R}(N)$. This inequivalence arises because in $I_D(N)$ the option to do an a -action can be disabled already before any visible action takes place; this is not possible in N .

The only possible way to avoid a distributed conflict in this net is by taking $t \equiv_D p \equiv_D u$. This is not allowed for any $D \in \mathcal{Q}_{\text{FD}}$ or $D \in \mathcal{Q}_{\text{SD}}$, but it is allowed for $D \in \mathcal{Q}_{\text{AD}}$ (cf. the bottom right net in Figure 3.1). Hence N is asymmetrically asynchronous, but not symmetrically asynchronous.

Since $\mathcal{Q}_{\text{FD}} \subseteq \mathcal{Q}_{\text{SD}} \subseteq \mathcal{Q}_{\text{AD}}$, any fully asynchronous net is symmetrically asynchronous, and any symmetrically asynchronous net is also asymmetrically asynchronous. Below we give semi-structural characterisations of these three classes of nets. The first two stem from [GGS08c], where the class of fully asynchronous nets is called $FA(B)$ and the class of symmetrically asynchronous nets is called $SA(B)$. The class $AA(B)$ in [GGS08c] is somewhat larger than our class of asymmetrically asynchronous nets, for it is based on a slightly more involved protocol for replacing synchronous by asynchronous interaction.

Definition 30 An unlabelled net $N = (S, T, F, M_0, \ell)$ has a

- *partially reachable conflict* iff

$$\begin{aligned} \exists t, u \in T \exists p \in \bullet t \cap \bullet u. t \neq u \wedge \exists M \in [M_0]_N \exists k, l \in \mathbb{N}. \\ k \cdot \bullet t \leq M \wedge l \cdot \bullet u(p) \leq M(p) \wedge k \cdot \bullet t(p) \not\leq M(p) - l \cdot \bullet u(p), \end{aligned}$$

- *partially reachable N* iff

$$\begin{aligned} \exists t, u \in T \exists p \in \bullet t \cap \bullet u \exists q \in \bullet u. t \neq u \wedge p \neq q \wedge \exists M \in [M_0]_N \exists k, l \in \mathbb{N}. \\ k \cdot \bullet t \leq M \wedge l \cdot \bullet u(p) \leq M(p) \wedge k \cdot \bullet t(p) \not\leq M(p) - l \cdot \bullet u(p), \end{aligned}$$

- *left and right border reachable M* iff

$$\begin{aligned} \exists t, u, v \in T \exists p \in \bullet t \cap \bullet u, q \in \bullet u \cap \bullet v. t \neq u \wedge u \neq v \wedge p \neq q \wedge \\ \exists M_1 \in [M_0]_N \exists k, l \in \mathbb{N}. \\ k \cdot \bullet t \leq M_1 \wedge l \cdot \bullet u(p) \leq M_1(p) \wedge k \cdot \bullet t(p) \not\leq M_1(p) - l \cdot \bullet u(p) \wedge \\ \exists M_2 \in [M_0]_N \exists k, l \in \mathbb{N}. \\ k \cdot \bullet v \leq M_2 \wedge l \cdot \bullet u(q) \leq M_2(q) \wedge k \cdot \bullet v(q) \not\leq M_2(q) - l \cdot \bullet u(q). \end{aligned}$$

Observation 6 If an unlabelled net $N = (S, T, F, M_0, \ell)$ is 1-safe and has no arc-weights, it has a

- *partially reachable conflict* iff

$$\exists t, u \in T \exists p \in \bullet t \cap \bullet u. t \neq u \wedge \exists M \in [M_0]_N. \bullet t \leq M ,$$

- *partially reachable N* iff

$$\exists t, u \in T \exists p \in \bullet t \cap \bullet u. t \neq u \wedge |\bullet u| > 1 \wedge \exists M \in [M_0]_N. \bullet t \leq M ,$$

- *left and right border reachable M* iff

$$\exists t, u, v \in T \exists p \in \bullet t \cap \bullet u \exists q \in \bullet u \cap \bullet v. \begin{array}{l} t \neq u \wedge u \neq v \wedge p \neq q \wedge \\ \exists M_1, M_2 \in [M_0]_N. \bullet t \leq M_1 \wedge \bullet v \leq M_2 . \end{array}$$

□

In effect, these definitions combine Definition 28 with various possible choices of \mathcal{Q} .

Theorem 2 Let N be an unlabelled net.

- N is fully asynchronous iff it has no partially reachable conflict.
- N is symmetrically asynchronous iff it has no partially reachable N.
- N is asymmetrically asynchronous iff it has no left and right border reachable M.

Proof: Straightforward with Theorem 1.

□

In the theory of nets, there have been extensive studies on classes of nets with certain structural properties like *free choice nets* [BS83, Bes87] and *simple nets* [BS83], as well as extensions of these classes. They are closely related to the net classes defined here, but they are defined without taking reachability into account. For a comprehensive overview and discussion of the relations between those purely structurally defined net classes and our net classes see [GGS08c]. Restricted to unlabelled, 1-safe nets without dead transitions (meaning that every transition t satisfies the requirement $\exists M \in [M_0]_N. \bullet t \subseteq M$), Theorem 2 says that a net is fully asynchronous iff it is conflict-free in the structural sense (no shared preplaces), symmetrically asynchronous iff it is a free choice net and asymmetrically asynchronous iff it is simple.

Our asynchronous net classes are defined for unlabelled nets only. There are two approaches to lifting them to labelled nets. One is to postulate that whether a net is asynchronous or not has nothing to do with its labelling function, so that after replacing this labelling by the identity function one can apply the insights above. This way our structural characterisations (Theorems 1 and 2) apply to labelled nets as well. Another approach would be to apply the notion

of behavioural asynchrony of Definition 27 directly to labelled nets. This way more nets will be asynchronous, because in some cases a net happens to be equivalent to its asynchronous implementation in spite of a failure of structural asynchrony. This happens for instance if all transitions in the original net are labelled τ . Unlike the situation for unlabelled nets, the resulting notion of behavioural asynchrony will most likely be strongly dependent on the choice of the semantic equivalence relation between nets.

3.2 The Asynchronous Implementation

Given a net N and a distribution D on N , this section explores the properties of the D -based asynchronous implementation $I_D(N)$ of N , focussing on the relationship between $I_D(N)$ and N , and culminating in the proof of Theorem 1 of Section 3.1.

The following lemma shows how the D -based asynchronous implementation of a net N simulates the behaviour of N .

Lemma 3 Let $N = (S, T, F, M_0, \ell)$ be a net, $G \in \mathbb{N}^T$, $\sigma \in \text{Act}^*$ and M_1, M_2 markings of N .

- If $M_1 [G]_N M_2$ then $M_1 \xrightarrow{\tau}_{I_D(N)}^* [G]_{I_D(N)} M_2$.
- If $M_1 \xRightarrow{\sigma}_N M_2$ then $M_1 \xRightarrow{\sigma}_{I_D(N)} M_2$.

Proof: Assume $M_1 [G]_N M_2$. Then, by construction of $I_D(N)$,

$$M_1 \left[\sum_{t \in G} \{t_s \mid s \in \bullet t, s \not\equiv_D t\} \right]_{I_D(N)} [G]_{I_D(N)} M_2.$$

The first part of that execution can be split into a sequence of singleton transitions, all labelled τ . The second statement follows by a straightforward induction on the length of σ . \square

This lemma uses the fact that any marking of N is also a marking on $I_D(N)$. The reverse does not hold, so in order to describe the degree to which the behaviour of $I_D(N)$ is simulated by N we need to explicitly relate markings of $I_D(N)$ to those of N . This is in fact not so hard, as any reachable marking of $I_D(N)$ can be obtained from a reachable marking of N by moving some tokens into the newly introduced buffering places s_t . To establish this formally, we can represent implementation markings as a pair of an original marking and a multiset of τ -transitions whose effects need to be applied to it. We notate the resulting pair as $M \oplus H$ to constantly remember that it actually denotes the marking $M + \llbracket H \rrbracket$. Addition and subtraction on such pairs are defined element-wise. We define a function which transforms implementation markings into such a form.

Definition 31 Let $N = (S, T, F, M_0, \ell)$ be an unlabelled net and let $I_D(N) = (S \cup S^\tau, T \cup T^\tau, F', M_0, \ell')$. $NF: \mathbb{N}^{S \cup S^\tau} \rightarrow \mathbb{Z}^S \times \mathbb{Z}^{T^\tau}$ is the function defined by

$$NF(M) := (M \upharpoonright S) + \sum_{s_t \in M \upharpoonright S^\tau} (\{s\} \oplus \{t_s\}) .$$

We now introduce a predicate α on pairs $M \oplus H$ and via NF on the markings of $I_D(N)$ that holds for a marking iff it can be obtained from a reachable marking of N (which is also a marking of $I_D(N)$) by firing some unobservable transitions. Each of these unobservable transitions moves a token from a place s into a buffering place s_t . Later, we will show that α exactly characterises the reachable markings of $I_D(N)$. Furthermore, as every token can be moved only once in this fashion, we can also give an upper bound on how many such movements can still take place.

Definition 32 Let $N = (S, T, F, M_0, \ell)$ be a finitary, unlabelled net and let $I_D(N) = (S \cup S^\tau, T \cup T^\tau, F', M_0, \ell')$. The predicate $\alpha \subseteq \mathbb{N}^S \times \mathbb{Z}^{T^\tau}$ is given by

$$\alpha(M \oplus H) \text{ iff } M \in [M_0]_N \wedge M + \llbracket H \rrbracket \in \mathbb{N}^{S \cup S^\tau} .$$

The function $d: \mathbb{N}^S \times \mathbb{Z}^{T^\tau} \rightarrow \mathbb{Z} \cup \{\infty\}$ is given by

$$d(M \oplus H) := |M \upharpoonright \{s \mid s \in S, \exists t \in s^\bullet . s \not\equiv_D t\}| - \sum_{t_s \in H} |\bullet t_s| ,$$

where we choose not to distinguish between different degrees of infinity.

The following lemma confirms that our informal description of α matches its formal definition.

Lemma 4 Let N and $I_D(N)$ be as above and $M \oplus H \in \mathbb{N}^S \times \mathbb{Z}^{T^\tau}$, with M finite. Then

1. $M + \llbracket H \rrbracket \in \mathbb{N}^{S \cup S^\tau}$ iff $M \xrightarrow{\tau}_{I_D(N)}^* M + \llbracket H \rrbracket$, and
2. $\alpha(M \oplus H) \Rightarrow d(M \oplus H) \in \mathbb{N}$.

Proof: (1): “If” follows from the definition of $\xrightarrow{\tau}$: For “only if”, note that $M \upharpoonright [H]_{I_D(N)} M + \llbracket H \rrbracket$ as ${}^\circ H \cap H^\circ = \emptyset$ by construction of $I_D(N)$.

(2): From $\alpha(M \oplus H)$ follows $M + \llbracket H \rrbracket \in \mathbb{N}^{S \cup S^\tau}$. Hence

$$\forall s \in S. M(s) \geq \sum_{t_s \in H} F'(s, t_s) .$$

As all members of H are from T^τ and by construction of the form t_s for some $t \in s^\bullet$ with $s \not\equiv_D t$, this ensures $\sum_{t_s \in H} |\bullet t_s| \leq |M \upharpoonright \{s \mid s \in S, \exists t \in s^\bullet . s \not\equiv_D t\}|$. \square

Now we can describe how any net simulates the behaviour of its fully asynchronous implementation.

Lemma 5 Let N and $I_D(N)$ be as above, $G \in \mathbb{N}^T$, $\sigma \in \text{Act}^*$ and $M, K \in \mathbb{N}^{S \cup S^\tau}$.

1. $\alpha(M_0 \oplus \emptyset)$.
2. If $\alpha(M \oplus H) \wedge M + \llbracket H \rrbracket [G]_{I_D(N)} K$
then $\exists M' \in \mathbb{N}^S, H' \in \mathbb{Z}^{T^\tau}. K = M' + \llbracket H' \rrbracket \wedge M[G]_N M' \wedge \alpha(M' \oplus H')$.
3. If $\alpha(M \oplus H) \wedge M + \llbracket H \rrbracket \xrightarrow{\tau}_{I_D(N)} K$
then $\exists M' \in \mathbb{N}^S, H' \in \mathbb{Z}^{T^\tau}. K = M' + \llbracket H' \rrbracket \wedge M = M' \wedge \alpha(M \oplus H') \wedge d(M \oplus H) > d(M \oplus H')$.
4. If $M_0 \xrightarrow{\sigma}_{I_D(N)} K$
then $\exists M' \in \mathbb{N}^S, H' \in \mathbb{Z}^{T^\tau}. K = M' + \llbracket H' \rrbracket \wedge M_0 \xrightarrow{\sigma}_N M' \wedge \alpha(M' \oplus H')$.

Proof: (1): $M_0 \in [M_0]_N$ and $M_0 \in \mathbb{N}^S$.

(2): A simple calculation shows that for any $t \in T$

$$\llbracket t \rrbracket_{I_D(N)} = \llbracket t \rrbracket_N - \left\llbracket \bigcup_{s \in \bullet t} \{t_s\} \right\rrbracket_{I_D(N)}. \quad (3.2)$$

Suppose $\alpha(M \oplus H)$ and $M + \llbracket H \rrbracket [G]_{I_D(N)} K$ with $G \in \mathbb{N}^T$. By definition of $[G]_{I_D(N)}$ we have $K = M + \llbracket H \rrbracket + \llbracket G \rrbracket_{I_D(N)}$. Applying (3.2), $K = M + \llbracket H \rrbracket + \llbracket G \rrbracket_N - \llbracket \sum_{t \in G} \bigcup_{s \in \bullet t} \{t_s\} \rrbracket_{I_D(N)}$. Reordering we take $M' := M + \llbracket G \rrbracket_N$ and $H' := H - \sum_{t \in G} \bigcup_{s \in \bullet t} \{t_s\}$.

From the definition of $[G]_{I_D(N)}$ we have $M' + \llbracket H' \rrbracket_{I_D(N)} \in \mathbb{N}^{S \cup S^\tau}$. To show $M[G]_N M'$ and thereby also $\alpha(M' \oplus H')$ we need $\bullet G \leq M$ and $M' = M + \llbracket G \rrbracket_N$.

The latter is true by definition. The former follows via Lemma 16 by the existence of the faithful path s, t_s, s_t, t with respect to $S_+ = S$ and $T_+ = T^\tau$ for each $t \in G$ and $s \in \bullet t$ with $s \not\equiv_D t$.

(3): From $M + \llbracket H \rrbracket \xrightarrow{\tau} K$ follows that $M + \llbracket H \rrbracket [t] K$ for some $t \in T \cup T^\tau$ with $\ell'(t) = \tau$ and $K = M + \llbracket H \rrbracket + \llbracket t \rrbracket$. We take $M' := M$ and $H' := H + \{t\}$. $M' = M$ and $K = M' + \llbracket H' \rrbracket$ follow immediately. From the definition of $\xrightarrow{\tau}$ follows $M' + \llbracket H' \rrbracket \in \mathbb{N}^{S \cup S^\tau}$ and as $M' = M$ we have $\alpha(M' \oplus H')$. Finally $d(M \oplus H) > d(M' \oplus H')$ as $H < H'$.

(4): Using (1–3), this follows by a straightforward induction on the number of transitions in the derivation $M_0 \xrightarrow{\sigma}_{I_D(N)} M'$. \square

In contrast to [GGS08a], the theory in this section employs the $M \oplus H$ notation I invented for [GGSU13]. The preceding proof of Lemma 5 was thereby shortened from the entire page it took in [GGS08a].

But let us continue towards Theorem 1, to which the following lemma is a crucial step.

Lemma 6 Let $N = (S, T, F, M_0, \ell)$ be an unlabelled, finitary net without a distributed conflict, w.r.t. a distribution D . Let $I_D(N) = (S \cup S^\tau, T \cup$

T^τ, F', M_0, ℓ' , $K_1 = M + \llbracket \emptyset \rrbracket$ with $M \in [M_0]_N$, and $K_1 \xrightarrow{\tau}_{I_D(N)} K_2 \xrightarrow{\tau}_{I_D(N)} \dots \xrightarrow{\tau}_{I_D(N)} K_n \xrightarrow{\tau}_{I_D(N)}$ for some $n \geq 1$. Let $G \in \mathbb{N}^T$.
Then $M[G]_N$ iff $K_n[G]_{I_D(N)}$.

Proof: By Lemma 5.3, we can write all K_i as $M + \llbracket H_i \rrbracket \in \mathbb{N}^{S \cup S^\tau}$

Suppose $G(t) = k$. Pick any $p \in \bullet t$. There are two cases to consider:

1. If $p \equiv_D t$ it follows that $k \cdot \circ t(p) \leq (M + \llbracket H_n \rrbracket)(p)$ as otherwise there must be some $u_p \in H_i$ which removed the tokens from p . From $p \equiv_D t$ follows $t_p \notin S^\tau$ and it would hold that $u \neq t \wedge p \not\equiv_D u$ and a distributed conflict with $l = H_n(u_p)$ would exist. This contradicts the assumptions.
2. If $p \not\equiv_D t$ it follows that $(M + \llbracket H_n \rrbracket)(p) < \bullet t(p)$ as otherwise $M + \llbracket H_n \rrbracket[t_p]_{I_D(N)}$ which contradicts $M + \llbracket H_n \rrbracket \xrightarrow{\tau}$. If $k \cdot \{t_p\} \not\leq M + \llbracket H_n \rrbracket$ this must be due to some $u_p \in H_n$. Taking $l = H_n(u_p)$ this constitutes a distributed conflict, violating the assumptions. Hence $k \cdot \{t_p\} \leq M + \llbracket H_n \rrbracket$.

In both cases $k \cdot \circ t \leq M + \llbracket H_n \rrbracket$.

It follows that $M_1[t]_N$ implies $M_n[t]_{I_D(N)}$ for all $t \in T$. Moreover, it follows immediately from the construction of $I_D(N)$ that if two transitions $t, u \in T$ are independent in N , then they are also independent in $I_D(N)$. Hence $M_1[G]_N$ implies $M_n[G]_{I_D(N)}$ for all $G \in \mathbb{N}^T$.

For the reverse direction, observe that $\alpha(M \oplus \emptyset)$, because $M \in [M_0]_N$. Hence $\alpha(M \oplus H_n)$ by Lemma 5.3 and $M + \llbracket H_n \rrbracket[G]_{I_D(N)}$ implies $M[G]_N$ by Lemma 5.2. \square

We can now prove Theorem 1 from earlier.

Theorem 1 Let $N = (S, T, F, M_0, \ell)$ be an unlabelled, finitary net, and \mathcal{Q} a requirement on distributions of nets.

N is behaviourally \mathcal{Q} -asynchronous iff it is structurally \mathcal{Q} -asynchronous.

Proof: “Only if”: Suppose N fails to be structurally \mathcal{Q} -asynchronous. Let D be a distribution on N meeting the requirement \mathcal{Q} . Then N has a distributed conflict with respect to D , i.e.

$$\begin{aligned} \exists t, u \in T \exists p \in \bullet t \cap \bullet u. t \neq u \wedge p \not\equiv_D u \wedge \exists M \in [M_0]_N \exists k, l \in \mathbb{N}. \\ k \cdot \bullet t \leq M \wedge l \cdot \bullet u(p) \leq M \wedge k \cdot \bullet t(p) \not\leq M(p) - l \cdot \bullet u(p). \end{aligned} \quad (3.3)$$

We need to show that $I_D(N) \not\approx_{\mathcal{Q}} N$.

Take the M, l, k, p, t, u from (3.3) and let $\sigma \in \text{Act}^*$ be such that $M_0 \xRightarrow{\sigma}_N M$. Then N has a step ready pair $\langle \sigma, X \rangle$ with $k \cdot \{\ell(t)\} \in X$. As unlabelled nets are deterministic [VN82], M is the only marking of N with the property that $M_0 \xRightarrow{\sigma}_N M$. Hence N has exactly one step ready pair after σ and it satisfies $k \cdot \{\ell(t)\} \in X$.

Lemma 3 yields $M_0 \xRightarrow{\sigma}_{I_D(N)} M$. Also $M[u_p]_{I_D(N)} M + \llbracket u_p \rrbracket$ by Definition 26, so $M \xrightarrow{\tau} M + \llbracket u_p \rrbracket$ and this process can be repeated l times. Let

$H_1 = l \cdot \{u_p\}$. Then $(M + \llbracket H_1 \rrbracket)(p) < k \cdot \bullet t(p)$. By Lemma 5.3, we have $M + \llbracket H_1 \rrbracket \xrightarrow{\tau}_{I_D(N)} M + \llbracket H_2 \rrbracket \xrightarrow{\tau}_{I_D(N)} \cdots \xrightarrow{\tau}_{I_D(N)} M + \llbracket H_n \rrbracket \not\xrightarrow{\tau}_{I_D(N)}$ for some $n \leq d(M \oplus \emptyset) \in \mathbb{N}$.

As $v^\circ \subseteq S^\tau$ for all $v \in T^\tau$, we have $(M + \llbracket H_i \rrbracket)(p) < k \cdot \bullet t(p)$ for $i = 1, 2, \dots, n$. Moreover, in case $p \not\equiv_D t$ we have $p_t \in v^\circ$ only if $v = \bullet t(p) \cdot \{p\}$; hence also $k \cdot \{p_t\} \not\leq M + \llbracket H_i \rrbracket$ for $i = 1, 2, \dots, n$. It follows that $k \cdot \circ t \not\leq M + \llbracket H_n \rrbracket$. Thus $I_D(N)$ has a step ready pair $\langle \sigma, X \rangle$ with $k \cdot \{\ell(t)\} \notin X$. We find that $\mathcal{R}(I_D(N)) \neq \mathcal{R}(N)$.

“If”: Suppose N is structurally \mathcal{Q} -asynchronous, i.e. there is a distribution D on N meeting the requirement \mathcal{Q} , such that N has no distributed conflicts with respect to D . We show that $\mathcal{R}(I_D(N)) = \mathcal{R}(N)$.

“ \supseteq ”: Let $\langle \sigma, X \rangle \in \mathcal{R}(N)$. Then there is a marking M of N such that $M_0 \xrightarrow{\sigma}_N M$ and, as N is unlabelled, for all $G \in \mathbb{N}^T$, $\ell(G) \in X \Leftrightarrow M[G]_N$. Lemma 3 yields $M_0 \xrightarrow{\sigma}_{I_D(N)} M$. By Lemma 5.3, we have $M \xrightarrow{\tau}_{I_D(N)} M + \llbracket H_1 \rrbracket \xrightarrow{\tau}_{I_D(N)} M + \llbracket H_2 \rrbracket \xrightarrow{\tau}_{I_D(N)} \cdots \xrightarrow{\tau}_{I_D(N)} M + \llbracket H_n \rrbracket \not\xrightarrow{\tau}_{I_D(N)}$ for some $0 \leq n \leq d(M \oplus \emptyset) \in \mathbb{N}$. Now Lemma 6 yields $M[G]_N \Leftrightarrow M + \llbracket H_n \rrbracket[G]_{I_D(N)}$. Hence $M + \llbracket H_n \rrbracket \xrightarrow{\ell(G)}_{I_D(N)} \Leftrightarrow \ell(G) \in X$ and $\langle \sigma, X \rangle \in \mathcal{R}(I_D(N))$.

“ \subseteq ”: Let $\langle \sigma, X \rangle \in \mathcal{R}(I_D(N))$. Then there is a marking M of $I_D(N)$ such that $M_0 \xrightarrow{\sigma}_{I_D(N)} M$, $M \not\xrightarrow{\tau}_{I_D(N)}$, and for each $A \in X$ there exists a unique $G \in \mathbb{N}^T$ with $\ell(G) = A$ such that $M[G]_{I_D(N)}$. Lemma 5.(4 and 3) yields $M_0 \xrightarrow{\sigma}_N M_1$ and some H_1 with $M_1 + \llbracket H_1 \rrbracket = M$ and $M_1 + \llbracket H_1 \rrbracket \not\xrightarrow{\tau}_{I_D(N)}$ and $\alpha(M_1 \oplus H_1)$ and Lemma 4 gives $M_1 \xrightarrow{\tau}_{I_D(N)} M_1 + \llbracket H_1 \rrbracket$. Now Lemma 6 yields $M[G]_{I_D(N)} \Leftrightarrow M_1[G]_N$ and thereby $\langle \sigma, X \rangle \in \mathcal{R}(N)$. \square

3.3 An Alternate Characterisation

The approach of Section 3.1 makes a difference between a net regarded as a specification, and an asynchronous implementation of the same net. The latter could be thought of as a way to execute the net when a given distribution makes the synchronisations that are inherent in the specification impossible. In this and the following section, on the other hand, we drop the difference between a net and its asynchronous implementation. Instead of adapting our intuition about the firing rule when implementing a net in a distributed way, we insist that all synchronisations specified in the original net remain present as synchronisations in a distributed implementation. Yet, at the same time we stick to the point of view that it is simply not possible for a transition to synchronise its firing with the removal of tokens from preplaces at remote locations. Thus we only allow distributions in which each transition is co-located with all of its preplaces. We call such distributions *effectual*. For effectual distributions D , the implementation transformation I_D is the identity. As a consequence, if effectuality is part of a requirement \mathcal{Q} imposed on distributions, the question whether a net is \mathcal{Q} -asynchronous is no longer dependent on whether an asynchronous implementation mimics the behaviour of the given net, but rather on whether the net allows a distribution satisfying \mathcal{Q} at all.

The requirement of effectuality does not combine well with the requirements on distributions proposed in Definition 25. For if \mathcal{Q} is the class of distributions that are effectual and asymmetrically distributed, then only nets without transitions with multiple preplaces would be \mathcal{Q} -asynchronous. This rules out most useful applications of Petri nets. The requirement of effectuality by itself, on the other hand, would make every net asynchronous, because we could assign the same location to all places and transitions.

We impose one more fundamental restriction on distributions, namely that when two visible transitions can occur in one step, they cannot be co-located. This is based on the assumption that at a given location visible actions can only occur sequentially, whereas we want to preserve as much concurrency as possible (in order not to lose performance). Recall that in Petri nets simultaneity of transitions cannot be enforced: if two transitions can fire in one step, they can also fire in any order. The standard interpretation of nets postulates that in such a case those transitions are causally independent, and this idea fits well with the idea that they reside at different locations.

3.3.1 Distributed Nets

This leads to the following definition of a distributed net.

Definition 33 [GGS08a] A net $N = (S, T, F, M_0, \ell)$ is *distributed* iff there exists a distribution D such that

- (1) $\forall s \in S, t \in T. s \in \bullet t \implies t \equiv_D s,$
- (2) $\forall t, u \in T. t \smile u \implies t \not\equiv_D u.$

A typical example of a net which is not distributed is shown in Figure 5.2 on Page 64. Transitions t and v are concurrently executable and hence should be placed on different locations. However, both have preplaces in common with u which would enforce putting all three transitions on the same location. In fact, distributed nets can be characterised in the following semi-structural way.

Observation 7 A net is distributed iff there is no sequence t_0, \dots, t_n of transitions with $t_0 \smile t_n$ and $\bullet t_{i-1} \cap \bullet t_i \neq \emptyset$ for $i = 1, \dots, n$. \square

Since a structural conflict net is defined as a net without such a sequence with $n = 1$ (cf. Definition 7), we obtain:

Observation 8 Every distributed net is a structural conflict net. \square

Further on, we use a more liberal definition of a distributed net, called *essentially distributed*. We will show that up to \approx_{bSTb}^Δ any essentially distributed net can be converted into a distributed net. In [GGS08a] we employed an even more liberal definition of a distributed net, which we call here *externally distributed*. Although we showed that up to step failures equivalence any externally distributed net can be converted into a distributed net, this does not hold for \approx_{bSTb}^Δ .

Definition 34 A net $N = (S, T, F, M_0, \ell)$ is *essentially distributed* iff there exists a distribution D satisfying (1) of Definition 33 and

$$(2') \quad \forall t, u \in T. t \sim u \wedge \ell(t) \neq \tau \implies t \not\equiv_D u.$$

It is *externally distributed* iff there exists a distribution D satisfying (1) and

$$(2'') \quad \forall t, u \in T. t \sim u \wedge \ell(t), \ell(u) \neq \tau \implies t \not\equiv_D u.$$

Instead of ruling out co-location of concurrent transitions in general, essentially distributed nets permit concurrency of internal transitions – labelled τ – at the same location. Externally distributed nets even allow concurrency between visible and silent transitions at the same location. If the transitions t and v in the net of Figure 5.2 would both be labelled τ , the net would be essentially distributed, although not distributed; in case only v would be labelled τ the net would be externally distributed but not essentially distributed. Essentially distributed nets need not be structural conflict nets; in fact, *any* net without visible transitions is essentially distributed.

Definition 35 Given any net N , the *canonical co-location relation* \equiv_C on N is the equivalence relation on the places and transitions of N generated by Condition (1) of Definition 33, i.e. the smallest equivalence relation \equiv_D satisfying (1). The *canonical distribution* of N is the distribution C that maps each place or transition to its \equiv_C -equivalence class.

Observation 9 A net that is distributed (resp. essentially or externally distributed) w.r.t. any distribution D , is distributed (resp. essentially or externally distributed) w.r.t. its canonical distribution.

This follows because whenever a co-location relation \equiv_D satisfies Condition (2) of Definition 33 (resp. Condition (2') or (2'') of Definition 34), then so does any smaller co-location relation. Hence a net is distributed (resp. essentially or externally distributed) iff its canonical distribution D satisfies (2) (resp. (2') or (2'')). \square

3.3.2 LSGA Nets

In this section we give an alternative, more concrete characterisation of nets representing distributed systems, by composing them asynchronously from sequential components. We call the resulting nets locally synchronous, globally asynchronous nets. We show below that the concrete characterisation of distributed systems as LSGA nets and the abstract characterisation as distributed nets agree. Formally, we introduce a model of distributed systems as nets consisting of component nets with sequential behaviour and interfaces in terms of input and output places.

Definition 36 Let $N = (S, T, F, M_0, \ell)$ be a net, $I, O \subseteq S$, $I \cap O = \emptyset$ and $O^\bullet = \emptyset$.

1. (N, I, O) is a *component with interface* (I, O) .
2. (N, I, O) is a *sequential component* with interface (I, O) iff

$$\exists Q \subseteq S \setminus (I \cup O) \text{ with } |M_0 \upharpoonright Q| = 1 \text{ and } \forall t \in T. |\bullet t \upharpoonright Q| = 1 \wedge |t^\bullet \upharpoonright Q| = 1.$$

An input place $i \in I$ of a component $\mathcal{C} = (N, I, O)$ can be regarded as a mailbox of \mathcal{C} for a specific type of messages. An output place $o \in O$, on the other hand, is an address outside \mathcal{C} to which \mathcal{C} can send messages. Moving a token into o is like posting a letter. The condition $o^\bullet = \emptyset$ says that a message, once posted, cannot be retrieved by the component.²

A set of places like Q above is a special case of an S -invariant. The requirements guarantee that the number of tokens in these places remains constant, in this case 1. It follows that no two transitions can ever fire concurrently. Conversely, whenever a net is sequential, in the sense that no two transitions can fire in one step, it is easily converted into a behaviourally equivalent net with the required S -invariant, namely by adding a single marked place with a self-loop to all transitions. This modification preserves virtually all semantic equivalences on nets from the literature, including \approx_{bSTb}^Δ .

Next we define an operator for combining components with asynchronous communication by fusing input and output places.

Definition 37 Let \mathfrak{K} be an index set.

Let $((S_k, T_k, F_k, M_{0k}, \ell_k), I_k, O_k)$ with $k \in \mathfrak{K}$ be components with interface such that $(S_k \cup T_k) \cap (S_l \cup T_l) = (I_k \cup O_k) \cap (I_l \cup O_l)$ for all $k, l \in \mathfrak{K}$ with $k \neq l$ (components are disjoint except for interface places) and $I_k \cap I_l = \emptyset$ for all $k, l \in \mathfrak{K}$ with $k \neq l$ (mailboxes cannot be shared; any message has a unique recipient).

Then the *asynchronous parallel composition* of these components is defined by

$$\parallel_{i \in \mathfrak{K}} ((S_k, T_k, F_k, M_{0k}, \ell_k), I_k, O_k) = ((S, T, F, M_0, \ell), I, O)$$

with $S = \bigcup_{k \in \mathfrak{K}} S_k$, $T = \bigcup_{k \in \mathfrak{K}} T_k$, $F = \bigcup_{k \in \mathfrak{K}} F_k$, $M_0 = \sum_{k \in \mathfrak{K}} M_{0k}$, $\ell = \bigcup_{k \in \mathfrak{K}} \ell_k$ (componentwise union of all nets), $I = \bigcup_{k \in \mathfrak{K}} I_k$ (we accept additional inputs from outside), and $O = \bigcup_{k \in \mathfrak{K}} O_k \setminus \bigcup_{k \in \mathfrak{K}} I_k$ (once fused with an input, $o \in O_k$ is no longer an output).

Note that the asynchronous parallel composition of components with interfaces is again a component with interface.

Observation 10 \parallel is associative.

This follows directly from the associativity of the (multi)set union operator. \square

We are now ready to define the class of nets representing systems of asynchronously communicating sequential components.

Definition 38 A net N is an *LSGA net* (a *locally sequential globally asynchronous net*) iff there exists an index set \mathfrak{K} and sequential components with interface \mathcal{C}_k , $k \in \mathfrak{K}$, such that $(N, I, O) = \parallel_{k \in \mathfrak{K}} \mathcal{C}_k$ for some I and O .

²We could have required that $\bullet I = \emptyset$, thereby disallowing a component to put messages in its own mailbox. This would not lead to a loss of generality in the class of distributed systems that can be obtained as the asynchronous parallel composition of sequential components, defined below. However, this property is not preserved under asynchronous parallel composition (defined below), and we like the composition of a set of (sequential) components to be a component itself (but not a sequential one).

Up to \approx_{bSTb}^Δ – or any reasonable equivalence preserving causality and branching time but abstracting from internal activity – the same class of LSGA systems would have been obtained if we had imposed, in Definition 36 of sequential components, that I , O and Q form a partition of S and that $\bullet I = \emptyset$.³ However, it is essential that our definition allows multiple transitions of a component to read from the same input place.

The idea of modelling asynchronously communicating sequential components by sequential nets interacting through buffer places has also been considered in [Rei82]. There Wolfgang Reisig introduces a class of systems, represented as nets, where the relative speeds of different components are guaranteed to be irrelevant. His class is a strict subset of our LSGA nets, requiring additionally, amongst others, that all choices in sequential components are free, i.e. do not depend upon the existence of buffer tokens, and that places are output buffers of only one component. Another quite similar approach was taken in [EHH10], where transition labels are classified as being either input or output. There, asynchrony is introduced by adding new buffer places during net composition. This framework does not allow multiple senders for a single receiver.

3.3.3 Relation between LSGA Nets and Distributed Nets

We proceed to show that the classes of LSGA nets, distributable nets and essentially distributable nets essentially coincide.

That every LSGA net is distributed follows because we can place each sequential component on a separate location. The following two lemmas constitute a formal argument. Here we call a component with interface (N, I, O) distributed iff N is distributed.

Lemma 7 Any sequential component with interface is distributed.

Proof: As a sequential component displays no concurrency, it suffices to co-locate all places and transitions. \square

Lemma 8 states that the class of distributed nets is closed under asynchronous parallel composition.

³First of all, any $i \in I$ with $\bullet i \neq \emptyset$ can be split into a pure input place, receiving tokens only from outside the component, and an internal place, which is the target of all arcs that used to go to i . Any transition t with $i \in \bullet t$ now needs to be split into one that takes its input token from the pure input place and one that takes it from the internal incarnation of i . In fact, if $F(i, t) = n$ then t needs to be split into $n+1$ copies. The result of this transformation is that $\bullet I = \emptyset$.

Next, any component $\mathcal{C} = ((S, T, F, M_0, \ell), I, O)$ with $\bullet I = \emptyset$ can be replaced by an equivalent component $((S', T', F', M'_0, \ell'), I, O)$ whose places S' are $I \dot{\cup} O \dot{\cup} Q$, where Q is the set of markings of \mathcal{C} , each restricted to the places outside I and O . For each transition t and markings M, M' of the component such that $M \xrightarrow{t} M'$, writing $q := M \upharpoonright (S \setminus (I \cup O))$ and $q' := M' \upharpoonright (S \setminus (I \cup O))$, there will be a transition $t_q \in T'$ with $F'(i, t_q) = F(i, t)$ for all $i \in I$, $F'(t_q, o) = F(t, o)$ for all $o \in O$, $F'(q, t) = F'(t, q') = 1$, and $F'(p, t) = F'(t, p) = 0$ otherwise. Moreover, $\ell'(t_q) = \ell(t)$ and M'_0 consists of the single place $M_0 \upharpoonright (S \setminus (I \cup O))$. This component clearly has the required properties.

Lemma 8 Let $\mathcal{C}_k = (N_k, I_k, O_k)$, $k \in \mathfrak{K}$, be components with interface, satisfying the requirements of Definition 37, which are all distributed. Then $\bigsqcup_{k \in \mathfrak{K}} \mathcal{C}_k$ is distributed.

Proof: We need to find a distribution D satisfying the requirements of Definition 33.

Every component \mathcal{C}_k is distributed and hence comes with a distribution D_k . Without loss of generality the codomains of all D_k can be assumed disjoint.

Considering each D_k as a function from net elements onto locations, a partial function D'_k can be defined which does not map any places in O_k , denoting that the element may be located arbitrarily, and behaves as D_k for all other elements. As an output place has no posttransitions within a component, any total function larger than (i.e. a superset of) D'_k is still a valid distribution for N_k .

Now $D' = \bigcup_{k \in \mathfrak{K}} D'_k$ is a (partial) function, as every place shared between components is an input place of at most one. The required distribution D can be chosen as any total function extending D' ; it satisfies the requirements of Definition 33 since the D_k 's do. \square

Corollary 1 Every LSGA net is distributed. \square

Corollary 2 Every LSGA net is a structural conflict net. \square

Conversely, any distributed net N , and even any essentially distributed net N , can be transformed in an LSGA net by choosing co-located transitions with their pre- and postplaces as sequential components and declaring any place that belongs to multiple components to be an input place of component N_k if it is a preplace of a transition in N_k , and an output place of component N_l if it is a postplace of a transition in N_l and not an input place of N_l . As transitions sharing a preplace are co-located, a place will be an input place of at most one component. Furthermore, in order to guarantee that the components are sequential in the sense of Definition 36, an explicit control place is added to each component – without changing behaviour – as explained below Definition 36. It is straightforward to check that the asynchronous parallel composition of all so-obtained components is an LSGA net, and that it is equivalent to N (using $\approx_{\mathcal{F}}$, \approx_{bSTb}^{Δ} , or any other reasonable equivalence).

Theorem 3 For any essentially distributed net N there is an LSGA net N' with $N' \approx_{bSTb}^{\Delta} N$.

Proof: Let $N = (S, T, F, M_0, \ell)$ be an essentially distributed net with a distribution D . Then an equivalent LSGA net N' can be constructed by composing sequential components with interfaces as follows.

For each equivalence class $[x]$ of net elements according to D a sequential component $(N_{[x]}, I_{[x]}, O_{[x]})$ is created. Each such component contains one new and initially marked place $p_{[x]}$ which is connected via self-loops to all transitions

in $[x]$. The interface of the component is formed by $I_{[x]} := (S \cap [x])^4$ and $O_{[x]} := ([x] \cap T)^\bullet \setminus [x]$. Formally, $N_{[x]} := (S_{[x]}, T_{[x]}, F_{[x]}, M_{0[x]}, \ell_{[x]})$ with

- $S_{[x]} = ((S \cap [x]) \cup O_{[x]} \cup \{p_{[x]}\})$,
- $T_{[x]} = T \cap [x]$,
- $F_{[x]} = F \upharpoonright (S_{[x]} \cup T_{[x]})^2 \cup \{(p_{[x]}, t), (t, p_{[x]}) \mid t \in T_{[x]}\}$,
- $M_{0[x]} = (M_0 \upharpoonright [x]) \cup \{p_{[x]}\}$, and
- $\ell_{[x]} = \ell \upharpoonright [x]$.

All components overlap at interfaces only, as the sole places not in an interface are the newly created $p_{[x]}$. The $I_{[x]}$ are disjoint as the equivalence classes $[x]$ are, so $(N', I', O') := \parallel_{[x] \in (S \cup T)/D} (N_{[x]}, O_{[x]}, I_{[x]})$ is well-defined. It remains to be shown that $N' \approx_{bSTb}^\Delta N$. The elements of N' are exactly those of N plus the new places $p_{[x]}$, which stay marked continuously except when a transition from $[x]$ is firing, and never connect two concurrently enabled transitions.

As we cannot have concurrently firing visible transitions on a single location, $|\overline{U} \cap [x]| \leq 1$ for any reachable ST-marking (M, U) of N and any $x \in S \cup T$, i. e. for any location $[x]$. Here \overline{U} is the multiset representation of the sequence U , defined in Section 2.2. The relation

$$\left\{ ((M, U), (M \cup S_U, U)) \mid \begin{array}{l} (M, U) \text{ is a reachable ST-marking of } N, \\ S_U = \{p_{[x]} \mid \overline{U} \cap [x] = \emptyset\} \end{array} \right\}$$

is a bijection between the reachable ST-markings of N' and N that preserves the ST-transition relations between them. In particular, if $(M, U) \xrightarrow{\tau} (M', U')$, using a silent transition that belongs to the equivalence class $[x]$, then $U' = U$ and $\overline{U'} \cap [x] = \emptyset$, i. e. no transition at location $[x]$ is currently firing, using that N is essentially distributed. Hence $p_{[x]} \in S_U$ and thus $(M \cup S_U, U) \xrightarrow{\tau} (M' \cup S_U, U)$. (This argument does not extend to externally distributed nets N .) From this it follows that $N' \approx_{bSTb}^\Delta N$. \square

Example 2 In Figure 5.1 appears an example of an essentially distributed net; the location borders are indicated. This net is not distributed, and thus not an LSGA net, because the two topmost τ -transitions are co-located but can be fired concurrently. Applying the construction in the proof of Theorem 3 turns this net into the distributed net of Figure 3.2.

Likewise, up to $\approx_{\mathcal{F}}$ any externally distributed net can be converted into a distributed net.

Proposition 4 [GGS08a] For any externally distributed net N there is a distributed net N' with $N' \approx_{\mathcal{F}} N$.

⁴Alternatively, we could take $I_{[x]} := (T \setminus [x])^\bullet \cap [x]$.

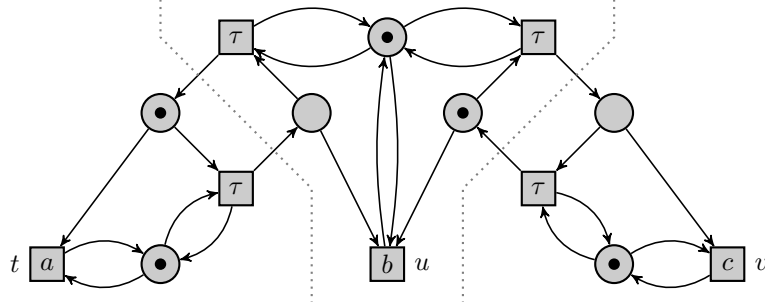


Figure 3.2: The LSGA net obtained from converting the essentially distributed net of Figure 5.1.

Proof: The same construction applies. The relation

$$\{(M, M \cup S) \mid M \text{ is a reachable marking of } N, S = \{p_{[x]} \mid [x] \text{ is a location}\}\}$$

is a bijection between the reachable markings of N' and N that preserves the step transition relations between them. Here we use that the transitions in the associated LTS involve either a multiset of concurrently firing *visible* transitions (that all reside on different locations and thus do not share a preplace $p_{[x]}$), or a single internal one. It follows that $N' \approx_{\mathcal{F}} N$. \square

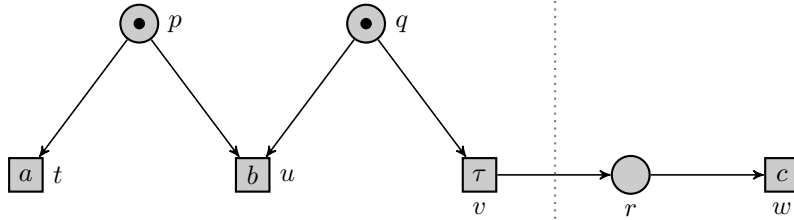


Figure 3.3: Externally distributed, but not convertible into a distributed net up to \approx_{bSTb}^{Δ} .

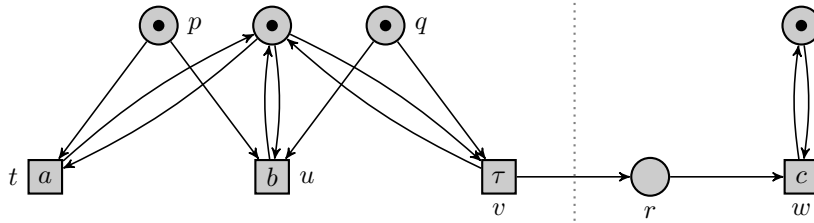


Figure 3.4: The LSGA net obtained from converting the externally distributed net of Figure 3.3.

Example 3 Figure 3.3 shows an externally distributed net; the (canonical) location borders are dotted. It is not essentially distributed, because the transitions t and v are co-located but can be fired concurrently, while $\ell(t) \neq \tau$. Applying the construction in the proof of Proposition 4 turns this net into the step failures equivalent LSGA net of Figure 3.4.

The counterexample in Figure 3.3 shows that up to \approx_{bSTb}^Δ not all externally distributed nets can be converted into distributed nets. Sequentialising the component with actions a , b and τ (as happens in Figure 3.4) would disable the execution $\xrightarrow{a^+} \xrightarrow{\tau} \xrightarrow{*} \xrightarrow{c^+}$.

Chapter 4

N-Free Nets and Related Classes

The potentially problematic nature of Ns was known long before we started our research. Hence a non-trivial subclass of nets – free choice nets – was identified, which disallowed Ns, gaining more efficient analysis algorithms in return. Theorem 2 provides a concrete and formal reason to dislike Ns on top of these earlier concerns. Naturally, we wished to gain a better understanding of the relationship between free choice nets and the various existing extensions thereof, and our class of symmetrically asynchronous nets.

This chapter presents the resulting joint work with Stephan Mennicke, published in [MSUG14].

4.1 Free-Choice and Extended Free-Choice

Free-choice nets (FC-nets) are generalizations of *S-nets* and *T-nets*, both representing net classes providing efficient analysis algorithms, e.g. for liveness or boundedness [Bes87, DE95, Esp98, BW13]. FC-nets allow choices between transitions, but only if there is at most one place in the preset of the conflicting transitions (Figure 4.1 (a)), and synchronization of places, but only in

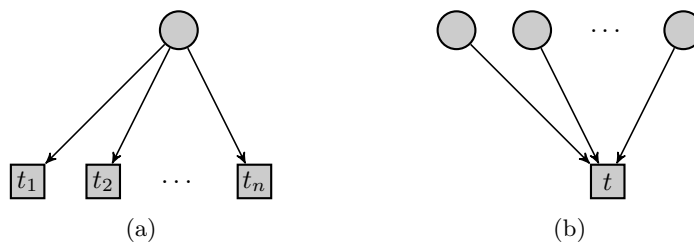


Figure 4.1: The two net principles allowed in FC-nets.

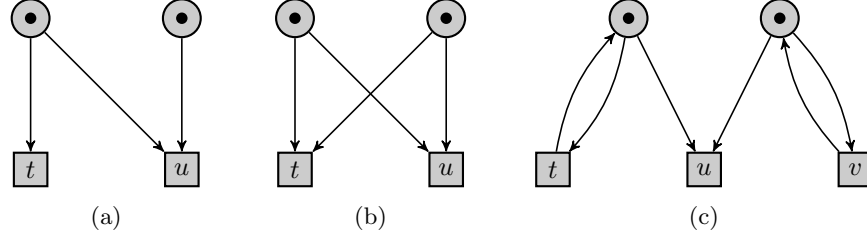


Figure 4.2: Three nets that are not FC-nets.

the way that there is at most one transition in the postset of the synchronized places (Figure 4.1 (b)). The conflict pattern refers to restrictions imposed on S-nets, while the synchronization pattern follows restrictions as imposed on T-nets. Hence, every time a conflict occurs, it occurs between the same set of transitions and it is not influenced by the rest of the net.

Definition 39 A net $N = (S, T, F, M_0, \ell)$ is a *free-choice net* (FC-net) iff $\forall p \in S \forall t \in T. F(p, t) > 0 \Rightarrow \exists k \in \mathbb{N}. p^\bullet = k \cdot \{t\} \vee {}^\bullet t = k \cdot \{p\}$.

Observation 11 A net $N = (S, T, F, M_0, \ell)$ without arc weights is free-choice iff $\forall p \in S \forall t \in T. p \in {}^\bullet t \Rightarrow p^\bullet = \{t\} \vee {}^\bullet t = \{p\}$. \square

FC-nets come with a natural notion of distributed component, called *free-choice cluster*. Given an FC-net $N = (S, T, F, M_0, \ell)$, the free-choice cluster of a transition $t \in T$ is the *conflict cluster* of t , i. e. $[t] = \{u \mid u \in ({}^\bullet t)^\bullet\}$. The set of all conflict clusters and thus free-choice clusters of N , denoted by $\mathcal{C}(N)$, imposes a partitioning on the set of transitions if N is an FC-net. Thus, a free-choice cluster may be seen as a component of a system N . Note that each component has at most one role, either to resolve a conflict between actions or to synchronise parallel components. Each component $C \in \mathcal{C}(N)$ is equipped with interfaces ${}^\bullet C$ (the set of input places) and C^\bullet (the set of output places). If some of the places of a component belong to the initial marking, these places are initially marked in the component. Hence, FC-nets are expressive enough to describe non-trivial distributed behaviour by a combination of locally restricted choices and synchronization. As already motivated, FC-nets excel in a wide range of applications, mostly due to their structural properties implying semantic properties. For a comprehensive overview, we refer to [Bes87, BW13]. Unfortunately, not every net is an FC-net (Figure 4.2). However, if a net is behaviourally equivalent to an FC-net, analysis results on the transformed net enables us to (partially) reason about semantic properties of the original net.

The net depicted in Figure 4.2 (b) belongs to the class of *extended free-choice nets* (EFC-nets). EFC-nets are characterized by the property that if two transitions share any place in their preset, then they share all preplaces. Thus, these nets exhibit the free-choice property to a certain extent, as conflicting transitions always depend on the same set of places.

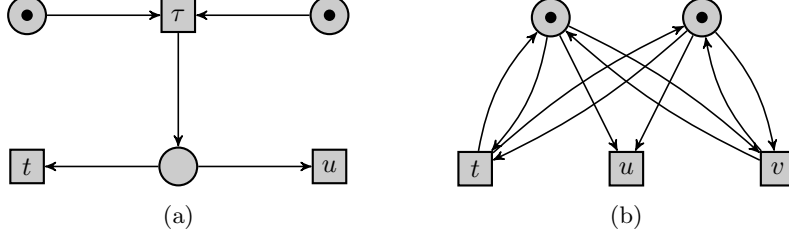


Figure 4.3: Transformation to FC-nets and EFC-nets, respectively.

Definition 40 A net $N = (S, T, F, M_0, \ell)$ is an *extended free-choice net* iff $\forall t_1, t_2 \in T. \bullet t_1 \cap \bullet t_2 \neq \emptyset \Rightarrow \bullet t_1 = \bullet t_2$.

Furthermore, we observe that the clustering described for FC-net transitions also works in EFC-nets, i.e. $\mathcal{C}(N)$ of an EFC-net N is a partition on the set of transitions of N . Therefore, we call a conflict cluster of an EFC-net *EFC-cluster*. Note that every FC-net is also an EFC-net, but the converse does not hold in general. However, Best and Shields proved that there is a behavioural correspondence by showing that EFC-nets may be transformed into FC-nets, respecting a form of *interleaving simulation* [BS83].

The core idea of that transformation is to split an EFC-cluster $C \in \mathcal{C}(N)$ into two operational parts, (i) a synchronization of all input places of C and (ii) a choice between all transitions in C . For (i), an unobservable transition is added, consuming the tokens from each input place of C and producing a token to an additional place p_C . In part (ii), all the transitions in C consume from p_C .

An example transformation, namely from the EFC-net in Figure 4.2 (b) into the corresponding FC-net is depicted in Figure 4.3 (a). Formally, the transformation is defined as follows [BS83, Bes87].

Definition 41 Let $N = (S, T, F, M_0, \ell)$ be an EFC-net. We then define $FC(N) := (S', T', F', M_0, \ell')$ where

$$\begin{aligned}
 S' &= S \cup \{p_{[u]} \mid [u] \in \mathcal{C}(N)\} \\
 T' &= T \cup \{\tau_{[u]} \mid [u] \in \mathcal{C}(N)\} \\
 F'(x, y) &= \begin{cases} F(x, y) & \text{if } x \in T \wedge y \in S \\ F(x, u) & \text{if } \exists u \in T. x \in S \wedge y = \tau_{[u]} \\ 1 & \text{if } \exists u \in T. x = \tau_{[u]} \wedge y = p_{[u]} \\ 1 & \text{if } y \in T \wedge x = p_{[y]} \\ 0 & \text{otherwise} \end{cases} \\
 \ell'(t) &= \begin{cases} \ell(t) & \text{if } t \in T \\ \tau & \text{otherwise} \end{cases}
 \end{aligned}$$

Following [BS83], $FC(N)$ is an FC-net simulating N if N is an EFC-net. The nature of the equivalence used in [BS83] however is not precisely specified. It

can be argued that this transformation from *EFC*-nets to *FC*-nets preserves branching time as well as causality, but disregards concurrency.

Let us now determine if *EFC*-nets remain equivalent to *FC*-nets when considering concurrency-aware equivalences. In *EFC*-nets, two transitions can only fire in a step if they do not share any preplace. Furthermore, the transformation defined above preserves conflict clusters in the sense that $\mathcal{C}(N) \subseteq \mathcal{C}(FC(N))$. Hence, if two transitions are concurrent in N , they still are so in $FC(N)$. Formally, the transformation function FC enjoys the property that an *EFC*-net N and $FC(N)$ are weak step bisimilar. The weak version of the equivalence is necessary, as FC introduces τ -transitions that do not belong to the original unlabelled *EFC*-net.

Theorem 4 Let N be an unlabelled, finitary *EFC*-net. Then $N \approx_{\mathcal{B}} FC(N)$.

Proof: Let $N = (S, T, F, M_0, \ell)$ be an unlabelled, finitary *EFC*-net and $N' = FC(N) = (S', T', F', M'_0, \ell')$. We prove $R = \{(M_1, M_2) \mid M_1 \in [N], M_1 \xrightarrow{\tau^*}_{N'} M_2\}$ to be a weak step bisimulation. Note that $S \subseteq S'$ and $T \subseteq T'$. Let $(M_1, M_2) \in R$. We need to check the following cases:

1. $(M_0, M'_0) \in R$, as $M_0 = M'_0$.
2. $M_1 \xrightarrow{A}_N M'_1$, i. e. $M_1[G]_N M'_1$ and $\ell(G) = A$. We also have $M_1 \xrightarrow{\tau^*}_{N'} M_2$. By Definition 41 for all $t \in T$ we have ${}^\circ\tau_{[t]} = \bullet t$, $\tau_{[t]}^\circ = {}^\circ t$, and $t^\circ = t^\bullet$. As the only τ -labelled transitions are of the form $\tau_{[t]}$ we conclude $M_2 = M_1 + \llbracket \sum_{t \in H} \{\tau_{[t]}\} \rrbracket$ for some $H \in \mathbb{N}^T$. If $(G \cup H) - H = \emptyset$ we take $M'_2 = M_2$, otherwise there is a M'_2 with $M_2[\sum_{t \in (G \cup H) - H} \{\tau_{[t]}\}]_{N'} M'_2$. In either case $M'_2[G]_{N'} M'_2$. And $M'_1[\sum_{t \in (G \cup H) - G} \{\tau_{[t]}\}]_{N'} M'_2$ or $(G \cup H) - G = \emptyset$ and $M'_1 = M'_2$. That $M_2 \xrightarrow{\tau^*}_{N'} \xrightarrow{A}_{N'} M'_2$ and $M'_1 \xrightarrow{\tau^*}_{N'} M'_2$ follows.
3. $M_1 \xrightarrow{\tau}_N M'_1$. As N is unlabelled, this case cannot occur.
4. $M_2 \xrightarrow{A}_{N'} M'_2$, i. e. $M_2[G]_{N'} M'_2$ and $\ell(G) = A$. We take $M'_1 = M_1 + \llbracket G \rrbracket_N$ and need to show that $M_1 \xrightarrow{A}_N M'_1$ and $(M'_1, M'_2) \in R$.
As before, we find $M_2 = M_1 + \llbracket \sum_{t \in H} \{\tau_{[t]}\} \rrbracket_{N'}$ for some $H \in \mathbb{N}^T$, which implies ${}^\circ \sum_{t \in H} \{\tau_{[t]}\} \leq M_1$. As ${}^\circ G = \sum_{t \in G} \{p_{[t]}\}$ and no $p_{[t]}$ exist in M_1 , we find $G \leq H$. As ${}^\circ \sum_{t \in G} \{\tau_{[t]}\} = \bullet G$ by the net construction, $\bullet G \leq M_1$ and $M_1[G]_N M'_1$. Then ${}^\circ \sum_{t \in (G + (H - G))} \{\tau_{[t]}\} \leq M_1$ gives $\bullet G + {}^\circ \sum_{t \in (H - G)} \{\tau_{[t]}\} \leq M_1$ and ${}^\circ \sum_{t \in (H - G)} \{\tau_{[t]}\} \leq M_1 - \bullet G + G^\bullet$ and hence $M_1[G]_N M'_1[\sum_{t \in (H - G)} \{\tau_{[t]}\}]_{N'} M'_2$ for some M'_2 . That $M'_2 = M'_2$ follows from $\llbracket \sum_{t \in H} \{\tau_{[t]}\} + G \rrbracket_{N'} = \llbracket G \rrbracket_N + \llbracket \sum_{t \in (H - G)} \{\tau_{[t]}\} \rrbracket_{N'}$ which can be seen from the net construction. Hence $(M'_1, M'_2) \in R$.
5. $M_2 \xrightarrow{\tau}_{N'} M'_2$. Trivially $M_1 \xrightarrow{\tau^*}_N M_1$ and $(M_1, M'_2) \in R$.

Hence, R is a weak step bisimulation between N and $FC(N)$. □

The proof of this theorem relies on the observation that the markings of N are also markings of $FC(N)$. The only difference is that the construction introduces some new places, one per EFC-cluster, that only hold a token if their preceding τ -transition has fired.

In conclusion, all EFC-nets can be simulated by FC-nets under concurrency-aware equivalences, using the well-known transformation. Intuitively, this paves the way for FC-net-based analysis for distributed systems being specified by EFC-nets. In the following section, we try to repeat this result for *behavioural free-choice nets* (BFC-nets). Unfortunately, not all BFC-nets exhibit (step) behaviour that may be expressed by an equivalent FC-net.

4.2 Behavioural Free-Choice

Free-choice may also be explored as a behavioural notion. The net in Figure 4.2 (c) is neither an FC-net nor an EFC-net. However, in every reachable marking, conflicting transitions t and u as well as u and v are either enabled or disabled. Although not all conflicting transitions depend on exactly the same resources, the initial conflict situation remains invariant. Once again, this meets the intuition of free-choice. This observation is the basis for *behavioural free-choice nets* (BFC-nets).

Definition 42 Let $N = (S, T, F, M_0, \ell)$ be a net. It is a *behavioural free-choice net* (BFC-net) if

$$\forall t, u \in T. \bullet t \cap \bullet u \neq \emptyset \Rightarrow \forall M \in [N]. M[\{t\}] \Leftrightarrow M[\{u\}] .$$

Every EFC-net is a BFC-net and thus every FC-net is also a BFC-net (as already [BS83] noted). However, for BFC-nets, the conflict clustering does not provide the property that $\mathcal{C}(N)$ partitions the set of transitions, e.g. the net depicted in Figure 4.2 (c) has two overlapping conflict clusters, namely $\{t, u\}$ and $\{u, v\}$. By merging those overlapping clusters, we reach a partitioning of the set of transitions, called BFC-clusters where $\langle u \rangle$ denotes the BFC-cluster of u . Thus, transitions in a BFC-cluster do not necessarily share all of their preplaces, e.g. t and v in Figure 4.2 (c) belong to the same BFC-cluster but $\bullet t \cap \bullet v = \emptyset$. However, the BFC-net property ensures that, e.g. in Figure 4.2 (c), both t and v are enabled as long as u does not fire.

In Figure 4.4 (a), another BFC-net is depicted. As before, either all transitions of the BFC-cluster $\langle t \rangle$ are enabled or disabled. Thus, without changing the (step) behaviour of the BFC-net, we may make each transition t dependent on all places in $\bullet \langle t \rangle$. In order to preserve the markings of the original net, each additional arc is complemented by an arc in the opposite direction. For the net in Figure 4.4 (a) we get the EFC-net depicted in Figure 4.4 (b). The same implementation idea applied to the BFC-net in Figure 4.2 (c) yields the EFC-net depicted in Figure 4.3 (b). For BFC-nets, the result of this construction is always an EFC-net that is interleaving branching time equivalent to the original net [Bes87]. Thus, by performing the FC-net construction described in Sect. 4.1,

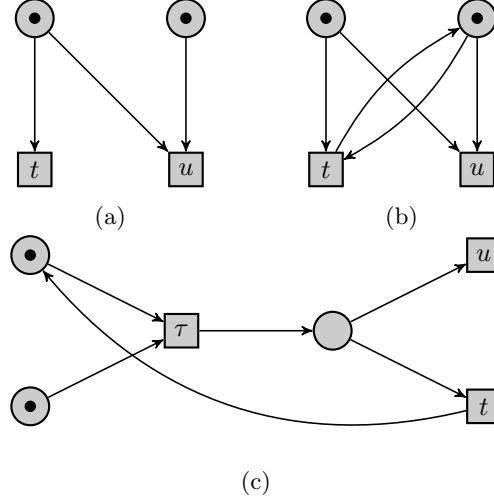


Figure 4.4: From BFC-nets to FC-nets.

we get an FC-net that is at least interleaving branching time equivalent to the original BFC-net.

Definition 43 Let $N = (S, T, F, M_0, \ell)$ be a net. We define a helper function $\alpha : S \times T \rightarrow \mathbb{N}$ as

$$\alpha(s, t) := \max_{u \in \langle t \rangle} \bullet u(s) .$$

Using α we define $EFC(N) := (S, T, F', M_0, \ell)$ where

$$F'(x, y) := \begin{cases} \alpha(x, y) & \text{if } x \in S \wedge y \in T \\ \alpha(y, x) - F(y, x) + F(x, y) & \text{if } x \in T \wedge y \in S \end{cases}$$

In Figure 4.4, we depicted all intermediate translation steps finally yielding the FC-net depicted in Figure 4.4 (c), which is indeed weak step bisimilar to the BFC-net in Figure 4.4 (a). Unfortunately, the EFC-net depicted in Figure 4.3 (b) is not weak step bisimilar to the net in Figure 4.2 (c), as the formerly concurrent transitions t and v are now conflicting. Hence, the proposed construction from the literature does not respect weak step bisimilarity.

In general, there is no transformation from the net in Figure 4.2 (c) into a step branching time equivalent (E)FC-net. This is due to the pattern, called *pure M*, the net contains, which will be properly introduced in Chapter 5 and shown to be stable under step branching time equivalences in Theorem 8. A *pure M* is characterized by three distinct transitions, t, u, v with $\bullet t \cap \bullet u \neq \emptyset$, $\bullet u \cap \bullet v \neq \emptyset$, and $\bullet t \cap \bullet v = \emptyset$ such that there is a reachable marking under which all three transitions are enabled. Thus, transitions t and v may fire concurrently while the synchronizing transition u is in conflict with both.

Obviously, the containment of *pure Ms* is no necessary condition for a net to be a BFC-net, e.g. the BFC-net in Figure 4.4 (a) does not even have three

transitions. Therefore, we devise two subclasses of BFC-nets that do not have pure Ms in the following. First, we restrict BFC-nets to those nets that do not have self-loops. Second, we look at the intersection of BFC-nets and so called asymmetric choice nets in which self-loops are allowed. See again [Bes87] for an overview on asymmetric choice nets. Both subclasses turn out to be step branching time equivalent to FC-nets.

4.2.1 BFC-nets without Self-Loops

In this section, we briefly discuss pure Ms in BFC-nets without self-loops. A self-loop is constituted by a transition t and a place p such that t consumes from and produces to p .

Definition 44 A net $N = (S, T, F, M_0, \ell)$ has a *self-loop* iff $\exists p \in S. \bullet p \cap p^\bullet \neq \emptyset$.

If we now consider BFC-nets without any self-loops, we need to reassess our argumentation above on why BFC-nets are not equivalent to FC-nets. Without a self-loop, a pure M is not expressible, because if t or v fired without returning the token to their preplaces they share with u , the overall net would not obey the BFC-net property. Indeed, the construction known from the literature works for BFC-nets without self-loops, proving that BFC-nets without self-loops are step-branching equivalent to (E)FC-nets.

Theorem 5 Let $N = (S, T, F, M_0, \ell)$ be an unlabelled BFC-net without self-loops. Then $EFC(N) \approx_B^A N$.

Proof: We take $R := \{(M, M) \mid M \in [M_0]\}$ and show it to be a weak step bisimulation with explicit divergence between $N = (S, T, F, M_0, \ell)$ and $EFC(N) = (S, T, F', M_0, \ell)$. We use \bullet to denote pre- and postsets in N and $^\circ$ when referring to $EFC(N)$.

1. Clearly $M_0 R M_0$.
2. Let $M \xrightarrow{A}_N M'$. Then $\exists G \in \mathbb{N}^T. \ell(G) = A \wedge M \geq \bullet G \wedge M' = M - \bullet G + G^\bullet$.
Take any $t \in G, s \in \bullet t$. From the construction of $EFC(N)$ we find that

$$\begin{aligned}
 t^\bullet(s) - \bullet t(s) &= F(t, s) - F(s, t) \\
 &= F(t, s) - F(s, t) + \alpha(s, t) - \alpha(s, t) \\
 &= (\alpha(s, t) - F(s, t) + F(t, s)) - \alpha(s, t) \\
 &= t^\circ(s) - \circ t(s)
 \end{aligned}$$

Hence $M' = M - \circ G + G^\circ$ and we just need to show that $M \geq \circ G$.

Assume the contrary. Then there is some place $s \in S$ with $M(s) < \circ G(s)$. First, consider individual $t \in s^\circ$ and assume $M(s) < \circ t(s) = \alpha(s, t)$. By definition of α then there must exist $u \in \langle t \rangle$ such that $\alpha(s, t) = \bullet u(s) > M(s)$. However, since N is a behavioural free choice net and $M[\langle t \rangle]_N$ all $u \in \langle t \rangle$ also have $M[\langle u \rangle]$ and thereby $\bullet u(s) \leq M(s)$. Hence there must be

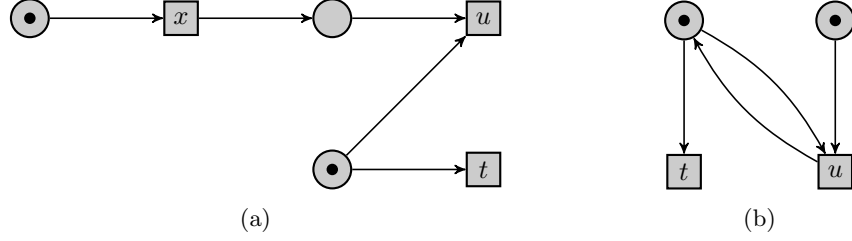


Figure 4.5: Two AC-nets having a partially and fully reachable N.

a non-singleton multiset G' of transitions from s° such that $M(s) < {}^\circ G'(s)$ yet $\bullet G' \leq M$. Then at least two transitions $t, u \in G', t \neq u$ exist with $u \in \langle t \rangle$. But from $M[\{u, t\}]$ follows $M[\{u\}]M''$ for some M'' and as N is behavioural free choice and t stays enabled, $M''[\{u, t\}]$. This implies u can be fired infinitely often, yet this is only possible if u has a self-loop, contradicting the assumptions.

3. As N is unlabelled, $\neg M \xrightarrow{\tau}_N$.
4. Let $M \xrightarrow{A}_{EFC(N)} M'$. Then $\exists G \in \mathbb{N}^T. \ell(G) = A \wedge M \geq {}^\circ G \wedge M' = M - {}^\circ G + G^\bullet$, i. e. we are left to show that $M \geq \bullet G$. Using the definition of α , ${}^\circ G \geq \bullet G$ and the claim follows.
5. As N and thereby $EFC(N)$ is unlabelled, $\neg M \xrightarrow{\tau}_{EFC(N)}$.
6. As N is unlabelled, $\neg M \xrightarrow{\tau}_N$.
7. As N and thereby $EFC(N)$ is unlabelled, $\neg M \xrightarrow{\tau}_{EFC(N)}$.

□

Next, we discuss a subclass that allows self-loops in BFC-nets but restricts the class structurally to not contain Ms.

4.2.2 BFC-nets and Asymmetric Choice

The condition for a net to be an EFC-net may be reformulated in terms of places, yielding exactly the same net class as by Definition 40 [Bes87]. In this definition, a net is an EFC-net iff for all places p and q it holds that $p^\bullet \cap q^\bullet \neq \emptyset$ implies that $p^\bullet = q^\bullet$. In an *asymmetric choice net* (AC-net), this restriction is weakened to an alternative between the two set-inclusions between p^\bullet and q^\bullet . Before discussing the combination of BFC-nets and AC-nets, we clarify the relation between AC-nets and FC-nets.

Definition 45 A net $N = (S, T, F, M_0, \ell)$ is an *asymmetric choice net* (AC-net) iff

$$\forall p, q \in S. p^\bullet \cap q^\bullet \neq \emptyset \Rightarrow p^\bullet \leq q^\bullet \vee q^\bullet \leq p^\bullet.$$

This restriction hardly enforces the concepts of free-choice, as it does not require conflicting transitions to be chosen freely. However, the net depicted in Figure 4.2 (a) shows an example AC-net, which may be simulated by an FC-net. We discuss this transformation in Sect. 4.3. Considering the same net in a larger context, it is unclear whether a token eventually occurs on the second input place of u . The AC-net in Figure 4.5 (a) shows an example context. The choice between transitions t and u does only occur if transition x fires in advance. In an adversary scenario, transition x might only fire if transition t has fired, which makes u unable to fire at all. In the literature, such a situation is called (*asymmetric*) *confusion* [RT86], which may not be simulated by any FC-net, as choices may never occur conditionally. In general, we characterise such conditional choices by the notion of *partially and fully reachable* Ns. Here, the existence of a subnet like the one in Figure 4.2 (a) is expected and two distinct markings, one under which t is enabled and one enabling t and u .

Definition 46 ¹ Let $N = (S, T, F, M_0, \ell)$ be a net. $(t, u) \in T^2$ is a *partially and fully reachable* N iff $\exists p \in \bullet t \cap \bullet u \wedge \exists q \in \bullet u, q \neq p$, and $\exists M_t, M \in [N]. \bullet t \leq M_t \wedge \bullet u \not\leq M_t \wedge \bullet t \leq M \wedge \bullet u \leq M$.

Note that this notion does not only characterise asymmetric confusion patterns, but also nets as the one depicted in Figure 4.5 (b). Just as in the case of pure Ms, partially and fully reachable Ns are stable w.r.t. step branching time equivalences.

Proposition 5 Let N be an unlabelled, finitary structural conflict net with a partially and fully reachable N. Then for all plain nets N' with $N \approx_{\mathcal{F}} N'$ it holds that N' has a partially and fully reachable N.

Proof: Let $N = (S, T, F, M_0, \ell)$ and (t, u) a partially and fully reachable N of N . Then there exist reachable markings M_t and M such that M_t enables t , but not u and M enables both. Thus, there is a σ such that $M_0 \xRightarrow{\sigma} M_t$ and since N is unlabelled, and hence deterministic [VN82], (1) $(\sigma, X) \in \mathcal{F}(N) \Rightarrow \{\ell(t)\} \notin X$ and (2) $(\sigma, X) \in \mathcal{F}(N) \Rightarrow \{\ell(u)\} \in X$. From the existence of M we deduce that there is a σ' such that $M_0 \xRightarrow{\sigma'} M$ and by determinism (3) $(\sigma', X') \in \mathcal{F}(N) \Rightarrow \{\ell(t)\} \notin X' \wedge \{\ell(u)\} \notin X'$. As N is a structural conflict net, (4) also $(\sigma', X') \in \mathcal{F}(N) \Rightarrow \{\ell(t), \ell(u)\} \in X'$.

Let $N' = (S', T', F', M'_0, \ell')$ be a net with $N \approx_{\mathcal{F}} N'$, i.e. $\mathcal{F}(N) = \mathcal{F}(N')$. Therefore, $\mathcal{F}(N')$ obeys (1)–(4). Hence, there exist transitions $t', u' \in T'$ with $\ell'(t') = \ell(t)$ and $\ell'(u') = \ell(u)$. By (3) and (4), we have some $p \in \bullet t' \cap \bullet u'$. By (1) and (2), we get that there is a place $q \in \bullet u' \setminus \bullet t'$. By (1), there is a marking $M_{t'}$ with $M'_0 \xRightarrow{\sigma} M_{t'}$ and $\bullet t' \leq M_{t'} \wedge \bullet u' \not\leq M_{t'}$, as otherwise there would be a failure pair $(\sigma, X) \in \mathcal{F}(N')$ with $\{\ell'(t')\} \in X$. By (3), there is a marking M' with $M'_0 \xRightarrow{\sigma'} M'$ with $\bullet t' \leq M' \wedge \bullet u' \leq M'$. By (4) $\{t', u'\}$ is no step from M' . Summarizing, N' has a partially and fully reachable N. \square

¹Note that partial reachability differs from how it is defined in Definition 30 where we don't care whether u is enabled. Hence the concept defined here is related, but maybe less so than suggested by its name.

Corollary 3 For some unlabelled finitary structural conflict AC-net, there exist no weak step failures equivalent plain FC-net.

However, the BFC-net in Figure 4.2 (c) does not respect the restrictions as imposed on AC-nets. In general, such structures may not occur in an AC-net and conversely, partially and fully reachable Ns are ruled out in BFC-nets, which paves the way for the following theorem.

Theorem 6 Let N be an unlabelled, finitary structural conflict BFC-net and an AC-net. Then $EFC(N) \approx_B^{\Delta} N$.

Proof: Let $N = (S, T, F, M_0, \ell)$ be a BFC-AC-net specification and $EFC(N) = N' = (S, T, F', M_0, \ell)$. We prove that $R = \{(M, M) \mid M \in [N]\}$ is a weak step bisimulation with explicit divergence between N and N' . By definition, $(M_0, M_0) \in R$. Let $(M_1, M_2) \in R$. The cases for τ -steps are obsolete, as N and N' are unlabelled. We again employ the $^\circ$ notation to denote pre-/postconditions in N' .

1. $M_1 \xrightarrow{A}_N M'_1$, i. e. there is a step G from M_1 to M'_1 and $\ell(G) = A$. As N' is τ -free, we have to give a marking M'_2 such that $M_2 \xrightarrow{A}_{N'} M'_2$ and $(M'_1, M'_2) \in R$. If G is enabled by M_2 , then $M'_2 = M'_1$ where $M_2[G]M'_2$, as $M_2 = M_1$ and tokens that are consumed due to the construction of N' are reproduced. Additionally, the original outgoing arcs of transitions are preserved. As N is a structural conflict net, for transitions $t_1, t_2 \in G$ it holds that $\bullet t_1 \cap \bullet t_2 = \emptyset$. It remains to be shown that G is enabled by M_2 , i. e. $^\circ G \leq M_2$. Let $t \in G$. As $F' \geq F$, it holds that $|\circ t| \geq |\bullet t|$. In case of equality t may fire under M_2 . If $|\circ t| > |\bullet t|$, then there are places $p, q \in S$ and a transition $u \in T$ such that $p \in \bullet t \cap \bullet u$, $q \notin \bullet t$ but $q \in \bullet u$ – due to the fact that N is an AC-net. Hence, in N' there is an additional arc from q to t . As N is a BFC-net, t is enabled iff u is enabled and hence, $q \in M_1$. Thus, $q \in M_2$. In conclusion $^\circ t \leq M_2$ for arbitrary $t \in G$, hence $^\circ G \leq M_2$, i. e. G is enabled by M_2 and may fire, producing a new marking M'_2 with the properties described above.
2. $M_2 \xrightarrow{A}_{N'} M'_2$, i. e. there is a step G from M_2 to M'_2 and $\ell(G) = A$. N is τ -free and therefore, we need to give a marking M'_1 such that $M_1 \xrightarrow{A}_N M'_1$ and $(M'_1, M'_2) \in R$. As before, a t in N has less or equal incoming arcs than t in N' . As, $M_2 = M_1$ by definition of R , G is a step from M_1 to a marking M'_1 . Every token that is consumed by some t in N' , due to the construction of N' , is reproduced in N' . In N such tokens are not even consumed by t . The original postsets of transitions are preserved and hence, $M_1[G]M'_1$ and $M'_2 = M'_1$ thus $(M'_1, M'_2) \in R$. \square

Note that Best and Shields already prove that EFC yields an EFC-net for BFC-nets [BS83], but for interleaving branching time only.

In [AKD98], Wil van der Aalst, Ekkart Kindler and Jörg Desel also introduce two extensions to asymmetric choice nets (called extended simple nets therein),

by allowing self-loops to ignore the discipline imposed by the asymmetric choice requirement. This however assumes a kind of “atomicity” of self-loops, which we did not allow in this thesis. In particular we do not implicitly assume that a transition will not change the state of a place it is connected to by a self-loop, since in case of deadlock, the temporary removal of a token from such a place might not be temporary indeed.

4.3 Symmetric Asynchrony

Throughout the last section, we have seen two subclasses of BFC-nets that are step branching time equivalent to FC-nets. However, BFC-nets do not represent a reasonable basis for the step branching time closure of FC-nets, as there are non-BFC-nets being behaviourally equivalent to FC-nets, e.g. the net depicted in Figure 4.2 (a).

In this section, we finally analyse the relation between symmetrically asynchronous nets and FC-nets, which we first conjectured in [GGS08c]. Given a symmetrically asynchronous net $N = (S, T, F, M_0, \ell)$, there is a symmetrical distribution $\lambda : S \cup T \rightarrow \mathcal{L}$ such that N has no distributed conflict w.r.t. λ . We transform N under λ by $FC_{sym}(N) = (S, T', F', M_0, \ell)$ where

- $T' := \{t \in T \mid t \text{ is live} \wedge \forall p \in \bullet t \forall u \in p^\bullet, u \text{ is live. } \lambda(p) = \lambda(u) \vee u = t\}$, and
- $F'(x, y) := F(x, y) \upharpoonright (S \times T' \cup T' \times S)$.

This transformation yields an FC-net by removing transitions obstructing the FC-net properties.

Lemma 9 Let N be a symmetrically asynchronous net. Then, $FC_{sym}(N)$ is an FC-net.

Proof: Let $N = (S, T, F, M_0, \ell)$ be a symmetrically asynchronous net and λ the required symmetrical distribution.

Let $N' = FC_{sym}(N)$ with set of transitions T' and arc relation F' . We use $^\circ$ to denote pre- and postsets in N' and show that for all $p \in S$ and all $t \in T'$, $(p, t) \in F'$ implies $\exists k \in \mathbb{N}. {}^\circ t = k \cdot \{p\} \vee p^\circ = k \cdot \{t\}$.

Assume there is a place p and a transition t with $(p, t) \in F'$, but $\nexists k \in \mathbb{N}. {}^\circ t = k \cdot \{p\} \vee p^\circ = k \cdot \{t\}$. Then, there exists a place $q \neq p$ with $q \in {}^\circ t$ and a transition $u \in T'$ with $t \neq u$ and $u \in p^\circ$. By construction ${}^\circ t \leq \bullet t \wedge {}^\circ p \leq \bullet p$. As λ is symmetrical, $\lambda(q) \neq \lambda(p) \neq \lambda(t)$, $\lambda(u) \neq \lambda(q) \neq \lambda(t)$, and hence $\lambda(t) \neq \lambda(u)$ (cf. Definition 25). There are two cases to consider, (1) $\lambda(p) = \lambda(u)$ and (2) $\lambda(p) \neq \lambda(u)$.

(1): $t \in T'$ hence t is live. As $p \in {}^\circ u \leq \bullet u$ and $t \in p^\circ \leq \bullet p$ and $t \neq u$ and $\lambda(t) \neq \lambda(p)$ the definition of T' ensures $u \notin T'$, contradicting a previous conclusion.

(2): $u \in T'$ hence u is live. As $u \in p^\circ \leq \bullet p$ and $p \in {}^\circ t \leq \bullet t$ and $t \neq u$ and $\lambda(u) \neq \lambda(p)$ the definition of T' ensures $t \notin T'$, contradicting $(p, t) \in F'$.

As in any case a contradiction is reached, $(p, t) \in F'$ indeed implies $\exists k \in \mathbb{N}. {}^\circ t = k \cdot \{p\} \vee p^\circ = k \cdot \{t\}$. Thus, $FC_{sym}(N)$ is an FC-net. \square

Fortunately, none of the removed transitions takes part in actual behaviour, as all of them are dead transitions, i.e. there is no reachable marking enabling them.

Lemma 10 Let $N = (S, T, F, M_0, \ell)$ be a symmetrically asynchronous net and $FC_{sym}(N) = (S, T', F', M_0, \ell)$. Then for all $t \in T \setminus T'$ there is no $M \in [M_0]_N$ such that $M[\{t\}]$.

Proof: Assume there is a $t \in T \setminus T'$ and a marking $M \in [M_0]_N$ such that t is enabled under M . By definition of T' , there is a $p \in \bullet t$ and a $u \in p^\bullet$ such that u is live, $\lambda(p) \neq \lambda(u)$, and $u \neq t$. As M enabled t , we have $\bullet t \leq M$. Consider two cases: $\bullet t(p) > \bullet u(p)$ and $\bullet t(p) \leq \bullet u(p)$.

Case $\bullet t(p) > \bullet u(p)$: As M enabled t , we have $\bullet t \leq M \wedge \bullet u(p) \leq \bullet t(p) \leq M(p)$ and with via integer arithmetic we find an l such that $l \cdot \bullet u(p) \leq M(p) \wedge \bullet t(p) \not\leq M(p) - l \cdot \bullet u(p)$, i.e. that a distributed conflict of N exists.

Case $\bullet t(p) \leq \bullet u(p)$: As u is live, there must be some M' with $M'[\{u\}]_N$. Hence we have $\bullet u \leq M' \wedge \bullet t(p) \leq \bullet u(p) \leq M'(p)$ and via integer arithmetic we find an l such that $l \cdot \bullet t(p) \leq M'(p) \wedge \bullet u(p) \not\leq M'(p) - l \cdot \bullet t(p)$, i.e. that a distributed conflict of N exists. \square

Hence, FC_{sym} yields step branching time equivalent FC-nets.

Theorem 7 Let N be an unlabelled symmetrically asynchronous net. Then $N \approx_{\mathcal{B}}^{\Delta} FC_{sym}(N)$.

This theorem follows from Lemma 10 and the observation that dead transitions can be removed from a net without changing behaviour. As every FC-net is also symmetrically asynchronous, we get the following correspondence between the step branching time closure of FC-nets and of symmetrically asynchronous nets.

Corollary 4 Let N be an unlabelled net. N is weak step bisimilar with explicit divergence to a plain FC-net iff N is weak step bisimilar with explicit divergence to a plain symmetrically asynchronous net.

4.4 Short Overview of the Results So Far

The results above give rise to the lattice in Figure 4.6, which concludes this chapter.

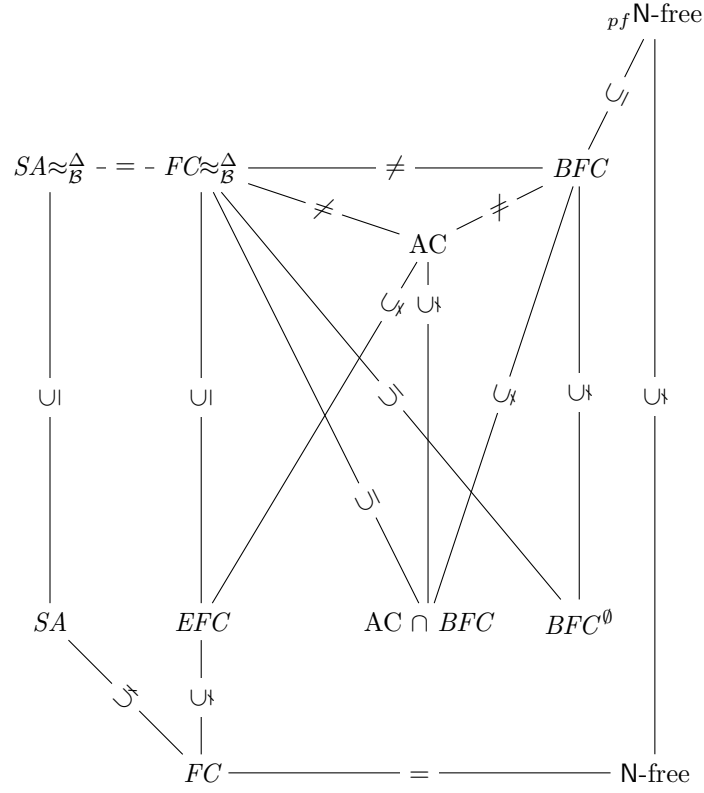


Figure 4.6: Most relationships from Chapters 3 and 4.

- SA : symmetrically asynchronous nets;
- $SA \approx_B^\Delta$: nets which are weakly bisimilar to nets in SA ;
- FC : free choice nets;
- EFC : extended free choice nets;
- $FC \approx_B^\Delta$: nets which are weakly bisimilar to nets in FC ;
- AC : asymmetric choice nets;
- BFC : behaviourally free choice nets;
- BFC^\emptyset : ... without self loops;
- pfN -free : nets without partially and fully reachable Ns.

Chapter 5

M-Free Nets and Related Classes

A step up from Ns are Ms, which will be the focus of the following chapter, repeating our work published in [GGSU13]. While Theorem 2 talks about left and right border reachable Ms, it will turn out that a different variant of Ms – fully reachable Ms – are in fact the more fundamental structure.

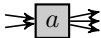

5.1 Reachable-M Free

In this chapter we consider nets as specifications of concurrent systems and ask the question which of those specifications can be implemented as distributed systems. This question can be formalised as

Which nets are semantically equivalent to distributed nets?

Of course the answer depends on the choice of a suitable semantic equivalence.

We will focus our attention to structural conflict nets and for them give a precise characterisation of those for which we can find semantically equivalent distributed nets. For the negative part of this characterisation, stating that certain nets are not distributable, we will use step failures equivalence, which is one of the simplest and least discriminating equivalences imaginable that abstracts from internal actions, but preserves branching time, concurrency and divergence to some small degree.¹ Giving up on any of these latter three properties would make any net distributable, but in a rather trivial and unsatisfactory way:

- Every net can be converted into an essentially distributed net by refining every transition  into the net segment .

¹In [GGSU11] we used *step readiness equivalence*, a slightly more discriminating equivalence with roughly the same properties. By moving to step failures equivalence we strengthen our result.

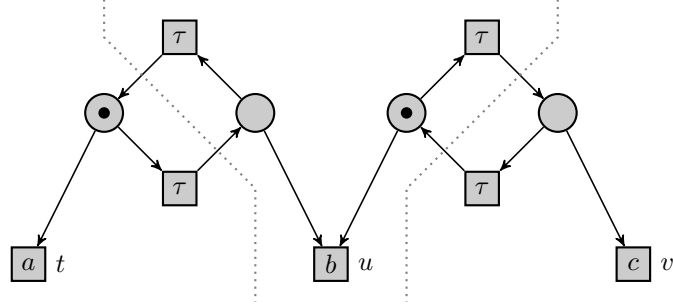


Figure 5.1: A busy-wait implementation of the net in Figure 5.2, location borders dotted.

This construction appears in [BD12] where it is criticised for putting “all relevant choice resolutions” on one location. The construction does not introduce or remove concurrency or divergence. So it preserves even causality respecting linear time equivalences like pomset trace equivalence [GG01]. It does not preserve branching time equivalences, because a choice between two visible transitions a and b in the original net is implemented by a choice between two internal transitions preceding a and b . The resulting net is essentially distributed because all new τ -transitions can be placed on the same location, whereas all other transitions get allocated a location of their own. Hence, using Theorem 3, it can be converted into an equivalent distributed net.

- When working in interleaving semantics, any net can be converted into an equivalent distributed net by removing all concurrency between transitions. This can be accomplished by adding a new, initially marked place, with an arc to and from every transition in the net.
- When fully abstracting from divergence, even when respecting causality and branching time, the net of Figure 5.2 is equivalent to the essentially distributed net of Figure 5.1, and in fact it is not hard to see that this type of implementation is possible for any given net. Yet, the implementation may diverge, as the non-deterministic choices might consistently be decided in an unhelpful way. This argument is elaborated in Section 5.1.1 below. The clause $M \not\rightarrow^\tau$ in Definition 11 is strong enough to rule out this type of implementation, even though our step failures semantics abstracts from other forms of divergence.

For the positive part, namely that all other nets are indeed distributable, we will use the most discriminating equivalence for which our implementation works, namely branching ST-bisimilarity with explicit divergence, which is finer than step failures equivalence. Hence we will obtain the strongest possible results for both directions and it turns out that the concept of distributability is fairly robust w.r.t. the choice of a suitable equivalence: any equivalence notion between

step failures equivalence and branching ST-bisimilarity with explicit divergence will yield the same characterisation.

Definition 47 A net N' is *distributable* up to an equivalence \approx iff there exists a distributed net N with $N \approx N'$.

Formally, we give our characterisation of distributability by classifying which finitary unlabelled structural conflict nets can be implemented as distributed nets, and hence as LSGA nets. In such implementations, we use invisible transitions. We study the concept “distributable” for unlabelled nets only, but in order to get the largest class possible we allow non-plain implementations, where a given transition may be split into multiple transitions carrying the same label.

5.1.1 Characterising Distributability

It is well known that sometimes a global protocol is necessary to implement synchronous interactions present in system specifications. In particular, this may be needed for deciding choices in a coherent way, when these choices require agreement of multiple components. The simple net in Figure 5.2 shows a typical situation of this kind. Independent decisions of the two choices might lead to incorrect system behaviour. If p and q both decide to send their respective tokens leftwards, a can fire, yet the token from q gets stuck as b never receives a second token. Compared to the correct semantics, a firing of c after a is missing. It can be argued that for this particular net there exists no satisfactory distributed implementation that fully respects the reactive behaviour of the original system: Transitions t and v are supposed to be concurrently executable (if we do not want to restrict performance of the system), and hence reside on different locations. Thus at least one of them, say t , cannot be co-located with transition u . However, both transitions are in conflict with u .

As we use nets as models of reactive systems, we allow the environment of a net to influence decisions at runtime by blocking some of the possibilities. Equivalently we can say it is the environment that fires transitions, and this can only happen for transitions that are currently enabled in the net. If the net decides between t and u before the actual execution of the chosen transition, the environment might change its mind in between, leading to a state of deadlock. Therefore we work in a branching time semantics, in which the option to perform t stays open until either t or u occurs. Hence the decision to fire u can only be taken at the location of u , namely by firing u , and similarly for t . Assuming that it takes time to propagate any message from one location to another, in no distributed implementation of this net can t and u be simultaneously enabled, because in that case we cannot exclude that both of them happen. Thus, the only possible implementation of the choice between t and u is to alternate the right to fire between t and u , by sending messages between them (cf. Figure 5.1). But if the environment only sporadically tries to fire t or u it may repeatedly miss the opportunity to do so, leading to an infinite loop of control messages sent back and forth, without either transition ever firing.

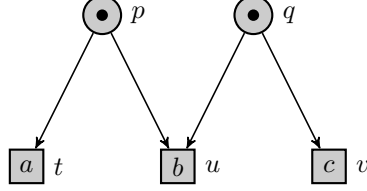


Figure 5.2: A fully reachable pure M.

Indeed such M-structures, representing interference between concurrency and choice, turn out to play a crucial rôle for characterising distributability. To be precise, it is only those Ms that are *pure*, i. e. don't have extra arcs from their places to their transitions besides those in Figure 5.2, and are *fully reachable*, i. e. for which there exists a reachable marking enabling all three transitions at the same time.

Definition 48 Let $N = (S, T, F, M_0, \ell)$ be a net. N has a *fully reachable visible pure M* iff
 $\exists t, u, v \in T. \bullet t \cap \bullet u \neq \emptyset \wedge \bullet u \cap \bullet v \neq \emptyset \wedge \bullet t \cap \bullet v = \emptyset \wedge \ell(t), \ell(u), \ell(v) \neq \tau \wedge$
 $\exists M \in [M_0]. \bullet t \cup \bullet u \cup \bullet v \leq M.$

Note that Definition 48 implies that $t \neq u$, $u \neq v$ and $t \neq v$.

Lemma 11 A net with a fully reachable visible pure M is not distributed.

Proof: Let $N = (S, T, F, M_0, \ell)$ be a net that has a fully reachable visible pure M, so there exist $t, u, v \in T$ and $p, q \in S$ such that $p \in \bullet t \cap \bullet u \wedge q \in \bullet u \cap \bullet v \wedge \bullet t \cap \bullet v = \emptyset$ and $\exists M \in [M_0]. \bullet t \cup \bullet u \cup \bullet v \leq M$. Then $t \sim v$. Suppose N is distributed by the distribution D . Then $t \equiv_D p \equiv_D u \equiv_D q \equiv_D v$ but $t \sim v$ implies $t \not\equiv_D v$. \nexists \square

We now give an upper bound on the class of distributable structural conflict nets by adapting a result from [GGS08a]: We show that fully reachable visible pure Ms that are present in an unlabelled structural conflict net are preserved under step failures equivalence. In [GGS08a] we showed this for step readiness equivalence.

Lemma 12 Let $N = (S, T, F, M_0, \ell)$ be an unlabelled structural conflict net. If N has a fully reachable visible pure M, then there are $\sigma \in \text{Act}^*$ and $a, b, c \in \text{Act}$ with $a \neq c$, such that $\langle \sigma, \{\{a, c\}\} \rangle, \langle \sigma, \{\{b\}\} \rangle \notin \mathcal{F}(N)$ and $\langle \sigma, \{\{a, b\}, \{b, c\}\} \rangle \in \mathcal{F}(N)$. (It is implied that $a \neq b \neq c$.)

Proof: N has a fully reachable visible pure M, so there exist $t, u, v \in T$ and $M \in [M_0]$ such that $\bullet t \cap \bullet u \neq \emptyset \wedge \bullet u \cap \bullet v \neq \emptyset \wedge \bullet t \cap \bullet v = \emptyset \wedge \bullet t \cup \bullet u \cup \bullet v \leq M$. Let $\sigma \in \text{Act}^*$ such that $M_0 \xrightarrow{\sigma} M$. Let $a := \ell(t)$, $b := \ell(u)$ and $c := \ell(v)$. Then $M \xrightarrow{\{a, c\}}$ and $M \xrightarrow{\{b\}}$. Moreover, using that N is a structural conflict net, $M \not\xrightarrow{\{a, b\}}$ and $M \not\xrightarrow{\{b, c\}}$. Since N is an unlabelled net, $M \not\xrightarrow{\tau}$, and there is no $M' \neq M$ with $M_0 \xrightarrow{\sigma} M'$. Hence $\langle \sigma, \{\{a, c\}\} \rangle, \langle \sigma, \{\{b\}\} \rangle \notin \mathcal{F}(N)$ and $\langle \sigma, \{\{a, b\}, \{b, c\}\} \rangle \in \mathcal{F}(N)$. \square

Lemma 13 Let $N = (S, T, F, M_0, \ell)$ be a structural conflict net. If there are $\sigma \in \text{Act}^*$ and $a, b, c \in \text{Act}$ with $a \neq c$, such that $\langle \sigma, \{\{a, c\}\} \rangle, \langle \sigma, \{\{b\}\} \rangle \notin \mathcal{F}(N)$ and $\langle \sigma, \{\{a, b\}, \{b, c\}\} \rangle \in \mathcal{F}(N)$, then N has a fully reachable visible pure \mathbf{M} .

Proof: Let $M \in \mathbb{N}^S$ be the marking that gives rise to the step failure pair $\langle \sigma, \{\{a, b\}, \{b, c\}\} \rangle$, i.e. $M_0 \xrightarrow{\sigma} M$, $M \not\xrightarrow{\{a, b\}}$ and $M \not\xrightarrow{\{b, c\}}$. Since $\langle \sigma, \{\{a, c\}\} \rangle \notin \mathcal{F}(N)$, it must be that $M \xrightarrow{\{a, c\}}$. Likewise, $M \xrightarrow{\{b\}}$.

As $a \neq b \neq c \neq a$ there must exist three transitions $t, u, v \in T$ with $\ell(t) = a \wedge \ell(u) = b \wedge \ell(v) = c$ and $M[\{t, v\}] \wedge M[\{u\}] \wedge \neg(M[\{t, u\}]) \wedge \neg(M[\{u, v\}])$. From $M[\{t, v\}] \wedge M[\{u\}]$ it follows that $\bullet t \cup \bullet u \cup \bullet v \leq M$ and $\bullet t \cap \bullet v = \emptyset$, using that N is a structural conflict net. From $\neg(M[\{t, u\}])$ then follows $\bullet t \cap \bullet u \neq \emptyset$ and analogously for u and v . Hence N has a fully reachable visible pure \mathbf{M} . \square

Note that the lemmas above give a behavioural property that for unlabelled structural conflict nets is equivalent to having a fully reachable visible pure \mathbf{M} .

Theorem 8 Let N be an unlabelled structural conflict net. If N has a fully reachable visible pure \mathbf{M} , then

1. any structural conflict net N' which is step failures equivalent to N has a fully visible pure \mathbf{M} , and hence
2. N is not distributable up to step failures equivalence.

Proof: (1): Let N be an unlabelled structural conflict net which has a fully reachable visible pure \mathbf{M} . Let N' be a net which is step failures equivalent to N . By Lemma 12 and Lemma 13, also N' has a fully reachable visible pure \mathbf{M} .

(2): By Lemma 11, N' is not distributed. Thus N is not distributable up to step failures equivalence. \square

Since \approx_{bSTb}^Δ is finer than $\approx_{\mathcal{F}}$, this result holds also for distributability up to \approx_{bSTb}^Δ (and any equivalence between $\approx_{\mathcal{F}}$ and \approx_{bSTb}^Δ).

In the following, we establish that the upper bound is tight, and hence a finitary unlabelled structural conflict net is distributable *iff* it has no fully reachable visible pure \mathbf{M} .

For this, it is helpful to first introduce macros for reversibility of transitions.

5.1.2 Nets with Reversible Transitions

A *net with reversible transitions* generalises the notion of a net; its semantics is given by a translation to an ordinary net, thereby interpreting the reversible transitions as syntactic sugar for certain net fragments. It is defined as a tuple $(S, T, \Omega, \iota, F, M_0, \ell)$ with S a set of places, T a set of (reversible) transitions, labelled by $\ell : T \rightarrow \text{Act} \dot{\cup} \{\tau\}$, Ω a set of *undo interfaces* with the relation $\iota \subseteq \Omega \times T$ linking interfaces to transitions, $M_0 \in \mathbb{N}^S$ an initial marking, and

$$F : (S \times T \times \{in, early, late, out, far\} \rightarrow \mathbb{N})$$

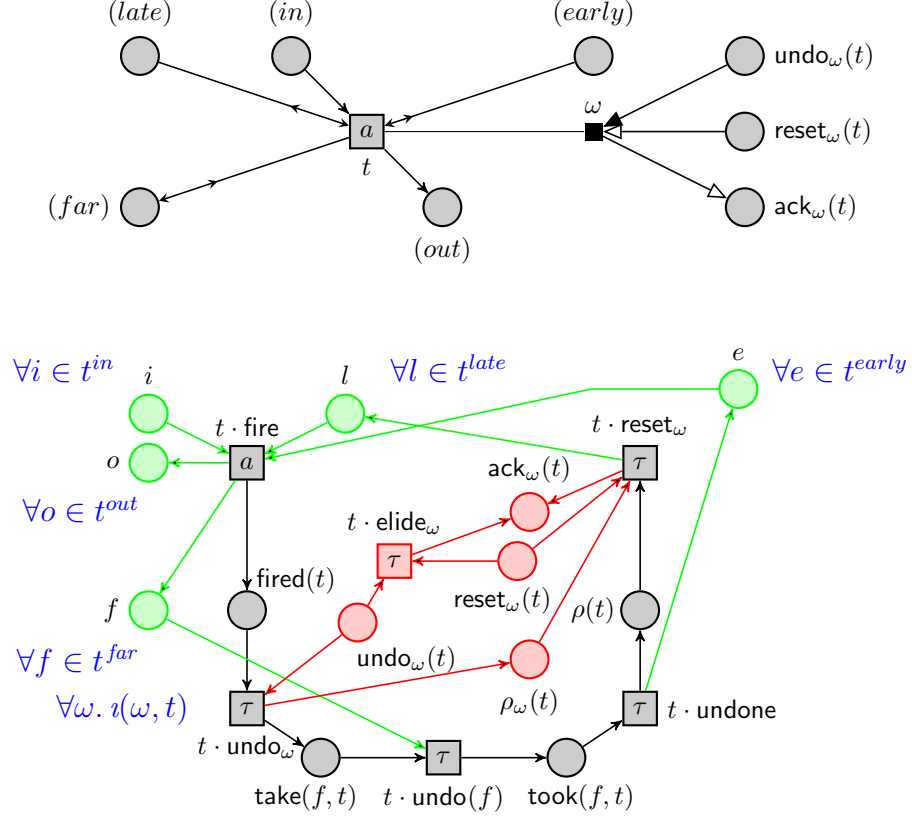


Figure 5.3: A reversible transition and its macro expansion.

the flow relation. When $F(s, t, type) > 0$ for $type \in \{in, early, late, out, far\}$, this is depicted by drawing an arc from s to t , labelled with its arc weight $F(s, t, type)$, of the form \longrightarrow , \longleftrightarrow , \longleftrightarrow , \longleftarrow , \longleftrightarrow , respectively. For $t \in T$ and $type \in \{in, early, late, out, far\}$, the multiset of places $t^{type} \in \mathbb{N}^S$ is given by $t^{type}(s) = F(s, t, type)$. When $s \in t^{type}$ for $type \in \{in, early, late\}$, the place s is called a *preplace* of t of type $type$; when $s \in t^{type}$ for $type \in \{out, far\}$, s is called a *postplace* of t of type $type$. For each undo interface $\omega \in \Omega$ and transition t with $\imath(\omega, t)$ there must be places $undo_\omega(t)$, $reset_\omega(t)$ and $ack_\omega(t)$ in S . A transition with a non-empty set of interfaces is called *reversible*; the other (*standard*) transitions may have pre- and postplaces of types *in* and *out* only – for these transitions $t^{in} = \bullet t$ and $t^{out} = t \bullet$. In case $\Omega = \emptyset$, the net is just a normal net.

A global state of a net with reversible transitions is given by a marking $M \in \mathbb{N}^S$, together with the state of each reversible transition “currently in progress”. Each transition in the net can fire as usual. A reversible transition can moreover take back (some of) its output tokens, and be *undone* and

reset. (The actual use in our implementation will be that every reversible transition that fires is undone and reset later.) When a transition t fires, it consumes $\sum_{type \in \{in, early, late\}} F(s, t, type)$ tokens from each of its preplaces s and produces $\sum_{type \in \{out, far\}} F(s, t, type)$ tokens in each of its postplaces s . A reversible transition t that has fired can start its reversal by consuming a token from $undo_\omega(t)$ for one of its interfaces ω . Subsequently, it can take back the tokens from its postplaces of type *far*. After it has retrieved all its output of type *far*, the transition is undone, thereby returning $F(s, t, early)$ tokens in each of its preplaces s of type *early*. Afterwards, by consuming a token from $reset_\omega(t)$, for the same interface ω that started the undo-process, the transition terminates its chain of activities by returning $F(s, t, late)$ tokens in each of its *late* preplaces s . At that occasion it also produces a token in $ack_\omega(t)$. Alternatively, two tokens in $undo_\omega(t)$ and $reset_\omega(t)$ can annihilate each other without involving the transition t ; this also produces a token in $ack_\omega(t)$. The latter mechanism comes in action when trying to undo a transition that has not yet fired, and gives us some flexibility when creating $undo_\omega(t)$ and $reset_\omega(t)$ tokens.

Figure 5.3 shows the translation of a reversible transition t with $\ell(t) = a$ into an ordinary net fragment. The arc weights on the green (or grey) arcs are inherited from the untranslated net; the other arcs have weight 1. Formally, a net $(S, T, \Omega, \iota, F, M_0, \ell)$ with reversible transitions translates into the net containing all places S , all standard transitions in T , labelled according to ℓ , along with their pre- and postplaces, and furthermore all net elements mentioned in Table 5.1, T^{\leftarrow} denoting the set of reversible transitions in T . The initial marking is exactly M_0 .

Transition	at	label	Preplaces	Postplaces	for all
$t \cdot \text{fire}$	t	$\ell(t)$	$t^{in}, t^{early}, t^{late}$	$\text{fired}(t), t^{out}, t^{far}$	$t \in T^{\leftarrow}$
$t \cdot \text{undo}_\omega$	$t\text{-undo}$	τ	$undo_\omega(t), \text{fired}(t)$	$\rho_\omega(t), \text{take}(f, t)$	$t \in T^{\leftarrow}, \iota(\omega, t), f \in t^{far}$
$t \cdot \text{undo}(f)$	f	τ	$\text{take}(f, t), f$	$\text{took}(f, t)$	$t \in T^{\leftarrow}, f \in t^{far}$
$t \cdot \text{undone}$	$t\text{-undo}$	τ	$\text{took}(f, t)$	$\rho(t), t^{early}$	$t \in T^{\leftarrow}, f \in t^{far}$
$t \cdot \text{reset}_\omega$	$t\text{-undo}$	τ	$\text{reset}_\omega(t), \rho_\omega(t), \rho(t)$	$t^{late}, \text{ack}_\omega(t)$	$t \in T^{\leftarrow}, \iota(\omega, t)$
$t \cdot \text{elide}_\omega$	$t\text{-undo}$	τ	$undo_\omega(t), \text{reset}_\omega(t)$	$\text{ack}_\omega(t)$	$t \in T^{\leftarrow}, \iota(\omega, t)$

Table 5.1: Expansion of a net with reversible transitions into a place/transition system.

A distribution of a net with reversible transitions can be given as a function $D : S \cup T \rightarrow \text{Loc}$. As in Condition (1) of Definition 33 we require that a transition and its preplaces (of types *in*, *early* or *late*) reside on the same location. Additionally, for any given transition t , all its undo-interface places $undo_\omega(t)$ and $reset_\omega(t)$ for all $\omega \in \Omega$ must reside on the same location – we refer to this location as $t\text{-undo}$. The second column of Table 5.1 indicates how such a distribution is translated under expansion of reversible transitions into ordinary net fragments: The location of a reversible transition t is really the location of $t \cdot \text{fire}$; it should be the same as all preplaces of t . Furthermore, the transition $t \cdot \text{undo}(f)$ and its preplace $\text{take}(f, t)$ reside on the same location as the place $f \in t^{far}$. All

other net elements that are part of the macro expansion of t , except for $\text{ack}_\omega(t)$, reside at the location $t\text{-undo}$. The resulting distribution of the expanded net is now guaranteed to satisfy (1). Whether a net with reversible translations is (essentially) distributed requires checking Condition (2) of Definition 33 (or Condition (2') of Definition 34) on its expansion.

5.1.3 The Conflict Replicating Implementation

Now we establish that a finitary unlabelled structural conflict net that has no fully reachable visible pure M is distributable. We do this by proposing the *conflict replicating implementation* of any such net, and show that this implementation is always (a) essentially distributed, and (b) equivalent to the original net. In order to get the strongest possible result, for (b) we use branching ST-bisimilarity with explicit divergence.

To define the conflict replicating implementation of $N' = (S', T', F', M'_0, \ell')$ we fix an arbitrary well-ordering $<$ on its transitions. We let $b, c, g, h, i, j, k, l, u$ range over these ordered transitions, and write

- $i \# j$ iff $i \neq j \wedge \bullet i \cap \bullet j \neq \emptyset$ (transitions i and j are *in conflict*), and $i \stackrel{\#}{=} j$ iff $i \# j \vee i = j$,
- $i <^\# j$ iff $i < j \wedge i \# j$, and $i \leq^\# j$ iff $i <^\# j \vee i = j$.

Figure 5.4 shows the conflict replicating implementation of N' . It is presented as a net

$$\mathcal{I}(N') = (S, T, F, \Omega, \iota, M_0, \ell)$$

with reversible transitions. The set Ω of undo interfaces is T' , and for $i \in \Omega$ we have $\iota(i, t)$ iff $t \in \Omega_i$, where the sets of transitions $\Omega_i \subseteq T$ are specified in Figure 5.4. The implementation $\mathcal{I}(N')$ inherits the places of N' (i.e. $S \supseteq S'$), and we define $M_0 \upharpoonright S'$ to be M'_0 . Given this, Figure 5.4 is not merely an illustration of $\mathcal{I}(N')$ – it provides a complete and accurate description of it, thereby defining the conflict replicating implementation of any net. In interpreting this figure it is important to realise that net elements are completely determined by their name (identity), and exist only once, even if they show up multiple times in the figure. For instance, the place $\pi_{h\#j}$ with $h=2$ and $j=5$ (when using natural numbers for the transitions in T') is the same as the place $\pi_{j\#l}$ with $j=2$ and $l=5$; it is a standard preplace of execute_2^i (for all $i \leq^\# 2$), a standard postplace of fetched_2^i , as well as a late preplace of transfer_5^2 . Figure 5.5 depicts the same net after expanding the macros for reversible transitions. An alternative description of the latter net appears in Table 5.2 on Page 88.

The rôle of the transitions distribute_p for $p \in S'$ is to distribute a token in p to copies p_j of p in the localities of all transitions $j \in T'$ with $p \in \bullet j$. In case j is enabled in N' , the transition initialise_j will become enabled in $\mathcal{I}(N')$. These transitions put tokens in the places pre_k^j , which are preconditions for all transitions execute_k^j , which model the execution of j at the location of k . When two conflicting transitions h and j are both enabled in N' , the first steps initialise_h and initialise_j towards their execution in $\mathcal{I}(N')$ can happen in parallel. To prevent them from executing both, execute_j^j (of j at its own location) is

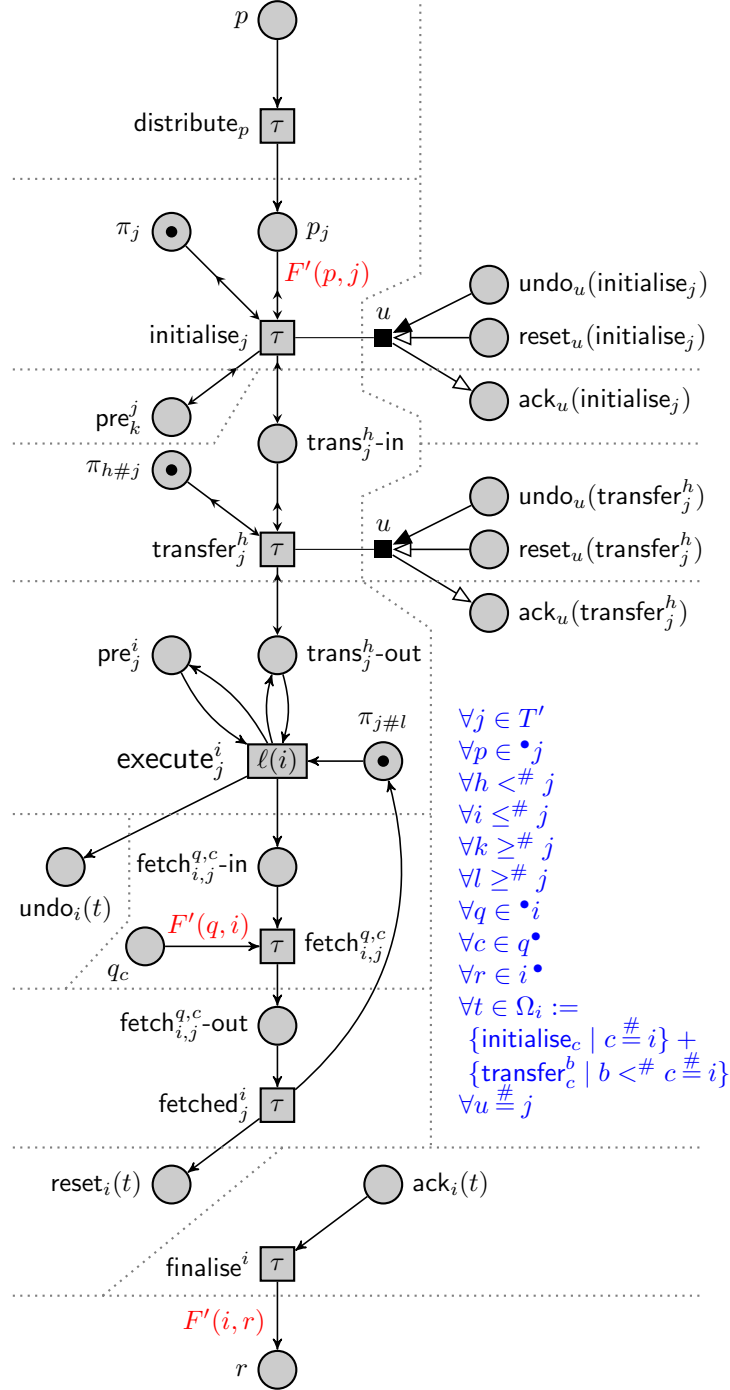


Figure 5.4: The entire conflict replicating implementation, drawn with emphasis on the structure of the component of j ; location borders dotted.

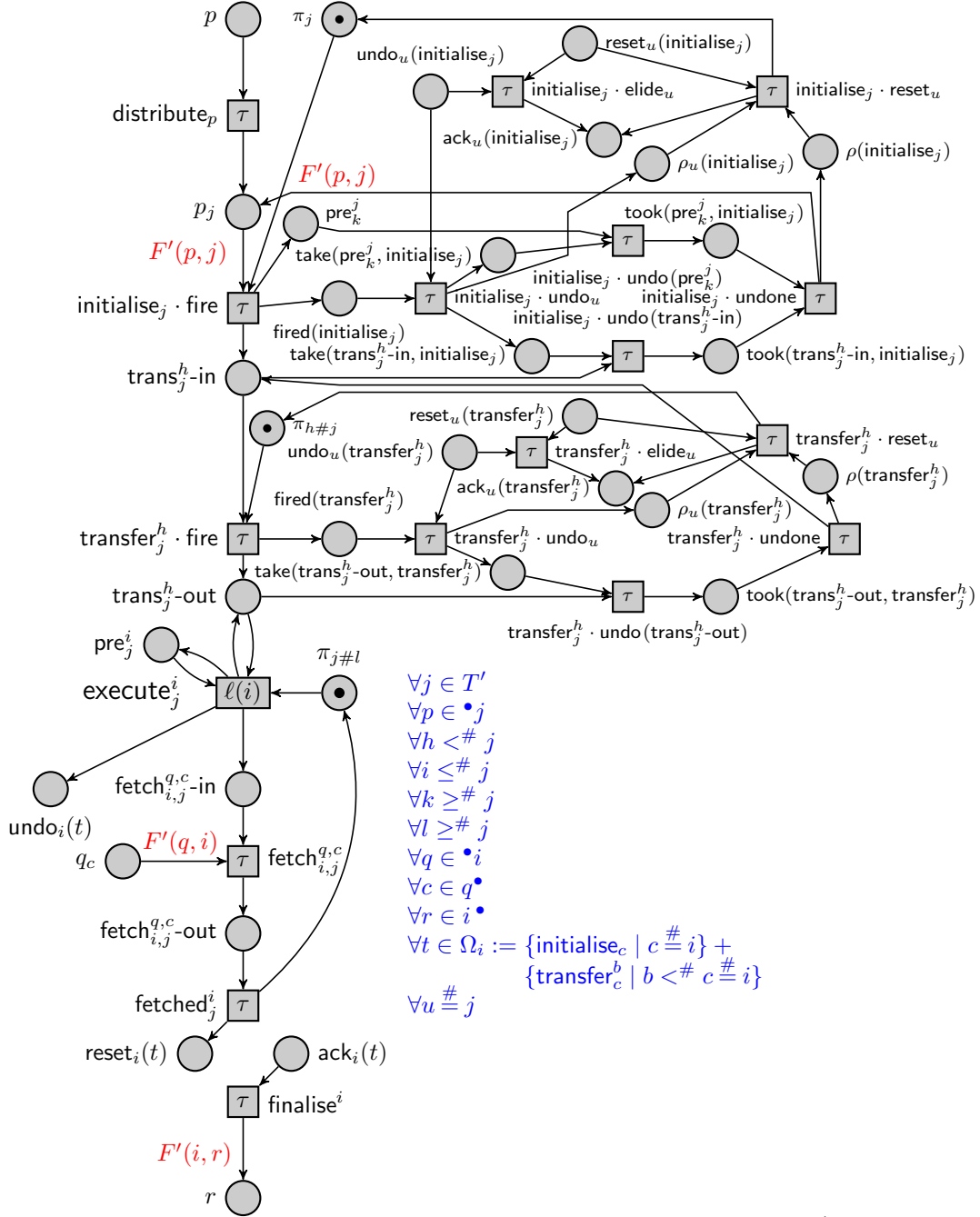


Figure 5.5: The entire conflict replicating implementation (with macros expanded).

only possible after transfer_j^h , which disables execute_h^h . This happens because transfer_j^h takes the initially present token from the place $\pi_{h\#j}$, which is needed to fire execute_h^h .

The main idea behind the conflict replicating implementation is that a transition $h \in T'$ is primarily executed by a sequential component of its own, but when a conflicting transition j gets enabled, the sequential component implementing j may “steal” the possibility to execute h from the home component of h , by putting a token in trans_j^h -in and getting transfer_j^h to fire, and then keep the options to do h and j open on the home component of j until one of them occurs. To prevent h and j from stealing each other’s initiative, which would result in deadlock, a global asymmetry is built in by ordering the transitions. Transition j can steal the initiative from h only when $h < j$.

In case j is also in conflict with a transition l , with $j < l$, the initiative to perform j may subsequently be stolen by l . In that case either h and l are in conflict too – then l takes responsibility for the execution of h as well – or h and l are concurrent – in that case h will not be enabled, due to the absence of fully reachable pure Ms in N' . The absence of fully reachable pure Ms also guarantees that it cannot happen that two concurrent transitions j and k both steal the initiative from an enabled transition h .

After the firing of execute_j^i all tokens that were left behind in the process of carefully orchestrating this firing will have to be cleaned up, in order to prepare the net for the next activity in the same neighbourhood. This is the reason for the reversibility of the transitions preparing the firing of execute_j^i . Hence there is an undo interface for each transition $i \in T'$, cleaning up the mess made in preparation of firing execute_j^i for some $j \geq^\# i$. Ω_i is the set of all transitions t that could possibly have contributed to this. For each of them the undo interface i is activated, by execute_j^i depositing a token in $\text{undo}_i(t)$. After all preparatory transitions that have fired are undone, tokens appear in the places p_c for all $p \in \bullet i$ and $c \in p^\bullet$. These are collected by $\text{fetch}_{i,j}^{p,c}$, after which all transitions in Ω_i get a reset signal. Those that have fired and were undone are reset, and those that never fired perform $\text{elide}_i(t)$. In either case a token appears in $\text{ack}_i(t)$. These are collected by finalise_i , which finishes the process of executing i by depositing tokens in its postplaces.

We allow multiple tokens to reside on the same place in the specification. To ensure that this does never lead to the component implementing a transition j starting the firing protocol again, even though it has not yet completed an earlier round, we introduce a place π_j which only holds a token while the component is idle.

By means of location boundaries, Figure 5.4 also displays a distribution of $\mathcal{I}(N')$. It has

- a location p for every place $p \in S'$, containing distribute_p and p ;
- locations initialise_j and execute_j for every $j \in T'$ – collectively referred to as “the location of j ” – the latter containing all transitions execute_j^i for $i \leq^\# j \in T'$;

- locations fetched_j^i for every $i \leq^\# j \in T'$;
- locations $\text{initialise}_j\text{-undo}$ for every $j \in T'$;
- locations $\text{transfer}_j^h\text{-undo}$ for every $h <^\# j \in T'$;
- and locations finalise_i for every $i \in T'$.

A transition transfer_j^h resides at location execute_h , due to its common preplace $\pi_{h\#j}$ with execute_h^g . Likewise, $\text{fetch}_{i,j}^{p,c}$ resides at location initialise_c . Provided N' is a finitary unlabelled structural conflict net without a fully reachable pure M , the proof of Theorem [12](#) will show that this distribution makes $\mathcal{I}(N')$ an essentially distributed net.

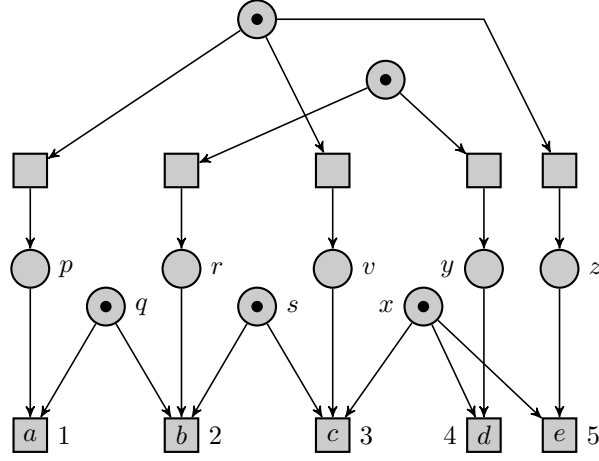


Figure 5.6: An example net.

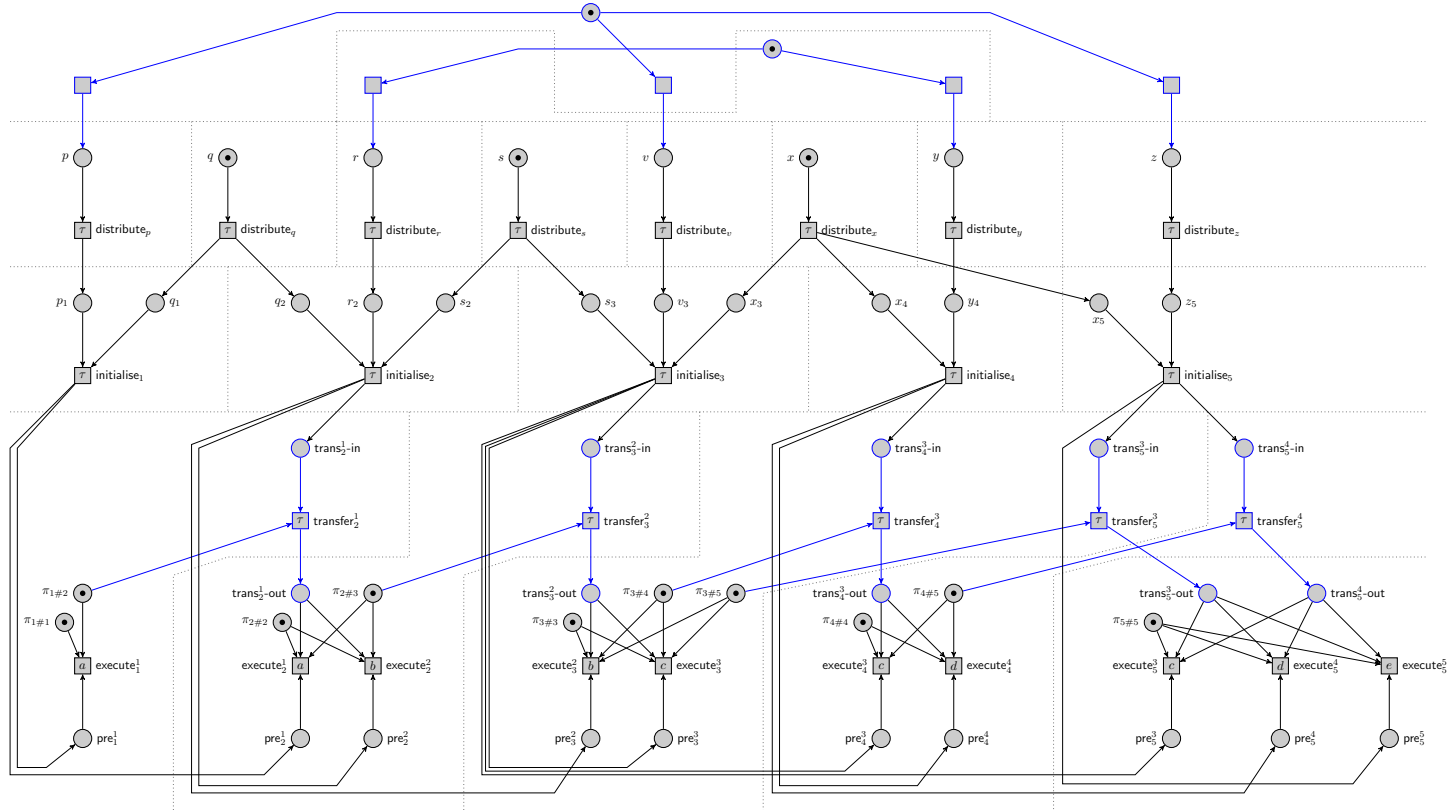


Figure 5.7: The (relevant parts of the) conflict replicating implementation of the net in Figure 5.6, location borders dotted.

The conflict replicating implementation is illustrated by means of the finitary unlabelled structural conflict net N' of Figure 5.6. The places and transitions $a-q-b-s-c-x-d$ in this net constitute a *Long M*: for each pair $a-b$, $b-c$ and $c-d$ of neighbouring transitions, as well as for the pair $a-d$ of extremal transitions, there exists a reachable marking enabling them both. Moreover, neighbouring transitions in the long *M* are in conflict: $a \# b$, $b \# c$ and $c \# d$, whereas the extremal transitions are concurrent: $a \smile d$. However, N' has no fully reachable pure *M*: no *M*-shaped triple of transitions $a-b-c$, $b-c-d$ or $b-c-e$ is ever simultaneously enabled.

In [GGS08a] we gave a simpler implementation, the *transition-controlled choice implementation*, that works for all finitary unlabelled 1-safe nets without such a long *M*. Hence N' constitutes an example where that implementation does not apply, yet the conflict replicating implementation does. In fact, when leaving out the $z-e$ -branch it may be the simplest example with these properties. We have added this branch to illustrate the situation where three transitions are pairwise in conflict.

Figure 5.7 presents relevant parts of the conflict replicating implementation $\mathcal{I}(N')$ of N' . What corresponds to the ten places of N' can easily be discerned in $\mathcal{I}(N')$, but the transitions of N' are replaced by more complicated net fragments. In Figure 5.7 we have simplified the rendering of $\mathcal{I}(N')$ by simply just copying the five topmost transitions of N' , instead of displaying the net fragments replacing them. This simplification is possible since the top half of N' is already distributed. To remind the reader of this, we left those transitions unlabelled.²

In order to fix a well-ordering $<$ on the remaining transitions, we named them after the first five positive natural numbers. The ordered conflicts between those transitions now are $1 \leq \#2$, $2 \leq \#3$, $3 \leq \#4$, $3 \leq \#5$ and $4 \leq \#5$. In Figure 5.7 we have skipped all places, transitions and arcs involved in the cleanup of tokens after firing of a transition. In this example the cleanup is not necessary, as no place of N' is visited twice. Thus, we displayed only the non-reversible part of the transitions initialise_j and transfer_j^h – i.e. $\text{initialise}_j \cdot \text{fire}$ and $\text{transfer}_j^h \cdot \text{fire}$ – as well as the transitions distribute_p and execute_j^i . Likewise, we omitted the outgoing arcs of execute_j^i , the places π_j , and those places that have arcs only to omitted transitions. We leave it to the reader to check this net against the definition in Figure 5.4, and to play the token game on this net, to see that it correctly implements N' .

In Section 5.3 we will show, for any finitary unlabelled structural conflict net N' without a fully reachable visible pure *M*, that $\mathcal{I}(N') \approx_{bSTb}^\Delta N'$, and that $\mathcal{I}(N')$ is essentially distributed. Hence $\mathcal{I}(N')$ is an essentially distributed implementation of N' . By Theorem 3 this implies that N' is distributable up to \approx_{bSTb}^Δ . Together with Theorem 8 it follows that, for any equivalence between $\approx_{\mathcal{F}}$ and \approx_{bSTb}^Δ ,

²While it is highly desirable in practical applications to use such simplifications to reduce the implementation size, we refrained from doing so in the formal definition of our implementation. It would have become less regular and the proofs correspondingly longer.

a finitary unlabelled structural conflict net is distributable iff it has no fully reachable visible pure M.

Given the complexity of our construction, no techniques known to us were adequate for performing the equivalence proof. We therefore had to develop an entirely new method for rigorously proving the equivalence of two nets up to \approx_{bSTb}^Δ , one of which known to be unlabelled. This method is presented in Section 5.2.

5.2 Proving Implementations Correct

This section presents a method for establishing the equivalence of two nets, one of which known to be unlabelled, up to branching ST-bisimilarity with explicit divergence. It appears as Theorem 9. First approximations of this method are presented in Lemmas 14 and 15. The progression from Lemma 14 to Lemma 15 and to Theorem 9 makes the method more specific (so less general) and more powerful. By means of a simplification a similar method can be obtained, also in three steps, for establishing the equivalence of two nets up to interleaving branching bisimilarity with explicit divergence. This is elaborated at the end of this section.

We sometimes illustrate the results of this section in terms of the conflict replicating implementation of a net defined in Section 5.1.3. However, the actual application of these results to show the correctness of that implementation is presented in Section 5.3.

Lemma 14 Let $N = (S, T, F, M_0, \ell)$ and $N' = (S', T', F', M'_0, \ell')$ be two finitary nets, N' being unlabelled. Suppose there is a relation $\mathcal{B} \subseteq (\mathbb{N}^S \times \mathbb{N}^T) \times (\mathbb{N}^{S'} \times \mathbb{N}^{T'})$ such that

- a) $(M_0, \emptyset)\mathcal{B}(M'_0, \emptyset)$,
- b) if $(M_1, U_1)\mathcal{B}(M'_1, U'_1)$ and $(M_1, U_1) \xrightarrow{\tau} (M_2, U_2)$ then $(M_2, U_2)\mathcal{B}(M'_1, U'_1)$,
- c) if $(M_1, U_1)\mathcal{B}(M'_1, U'_1)$ and $(M_1, U_1) \xrightarrow{\eta} (M_2, U_2)$ for some $\eta \in \text{Act}^\pm$ then $\exists(M'_2, U'_2). (M'_1, U'_1) \xrightarrow{\eta} (M'_2, U'_2) \wedge (M_2, U_2)\mathcal{B}(M'_2, U'_2)$,
- d) if $(M_1, U_1)\mathcal{B}(M'_1, U'_1)$ and $(M'_1, U'_1) \xrightarrow{\eta}$ with $\eta \in \text{Act}^\pm$ then either $(M_1, U_1) \xrightarrow{\eta}$ or $(M_1, U_1) \xrightarrow{\tau}$
- e) and there is no infinite sequence $(M, U) \xrightarrow{\tau} (M_1, U_1) \xrightarrow{\tau} (M_2, U_2) \xrightarrow{\tau} \dots$ with $(M, U)\mathcal{B}(M', U')$ for some (M', U') .

Then \mathcal{B} is a branching split bisimulation with explicit divergence, and $N \approx_{bSTb}^\Delta N'$.

Proof: It suffices to show that \mathcal{B} satisfies Conditions 1–7 of Definition 17; the condition on explicit divergence follows immediately from (e), using that an unlabelled net admits no divergence at all.

1. By (a).
2. By (c) with $\eta = a^+$.
3. By (c) with $\eta = a^-$.
4. By (b).
5. Suppose $\mathfrak{M}_1 \mathcal{B} \mathfrak{M}'_1$ and $\mathfrak{M}'_1 \xrightarrow{a^+} \mathfrak{M}'_2$. By (d) we have either $\mathfrak{M}_1 \xrightarrow{a^+} \mathfrak{M}_1^1$ or $\mathfrak{M}_1 \xrightarrow{\tau} \mathfrak{M}_1^1$ for some \mathfrak{M}_1^1 . In the latter case (b) yields $\mathfrak{M}_1^1 \mathcal{B} \mathfrak{M}_2$, and using (d) again, either $\mathfrak{M}_1^1 \xrightarrow{a^+} \mathfrak{M}_1^2$ or $\mathfrak{M}_1^1 \xrightarrow{\tau} \mathfrak{M}_1^2$ for some \mathfrak{M}_1^2 . Repeating this argument, if the choice between a^+ and τ is made k times in favour of τ (with $k \geq 0$), we obtain $\mathfrak{M}_1^k \mathcal{B} \mathfrak{M}_2$ (where $\mathfrak{M}_1^0 := \mathfrak{M}_1$) and either $\mathfrak{M}_1^k \xrightarrow{a^+} \mathfrak{M}_1^{k+1}$ or $\mathfrak{M}_1^k \xrightarrow{\tau} \mathfrak{M}_1^{k+1}$. By (e), at some point the choice must be made in favour of a^+ , say at \mathfrak{M}_1^k . Thus $\mathfrak{M}_1 \xrightarrow{\tau^*} \mathfrak{M}_1^k \xrightarrow{a^+} \mathfrak{M}_1^{k+1}$, with $\mathfrak{M}_1^k \mathcal{B} \mathfrak{M}_2$. We use \mathfrak{M}_1^{k+1} as the \mathfrak{M}_2 from Definition 17. It remains to show that $\mathfrak{M}_1^{k+1} \mathcal{B} \mathfrak{M}'_2$. By (c) there is an \mathfrak{M}'_3 with $\mathfrak{M}'_1 \xrightarrow{a^+} \mathfrak{M}'_3$ and $\mathfrak{M}_1^{k+1} \mathcal{B} \mathfrak{M}'_3$. Since N' is unlabelled, $\mathfrak{M}'_2 = \mathfrak{M}'_3$.
6. As the preceding case, substituting a^- for a^+ .
7. As N' is unlabelled, $\mathfrak{M}'_1 \xrightarrow{\tau}$ cannot occur.

The final conclusion follows by Proposition 2. \square

Lemma 14 provides a method for proving $N \approx_{bSTb}^{\Delta} N'$ that can be more efficient than directly checking the definition. In particular in Condition (d) one no longer has to match the targets of corresponding transitions. Lemma 15 below, when applicable, provides an even more efficient method: it is no longer necessary to specify the branching split bisimulation \mathcal{B} , and the targets have disappeared from the transitions in Condition bc as well. Instead, we have acquired Condition a, but this is a structural property, which is relatively easy to check.

Lemma 15 Let $N = (S, T, F, M_0, \ell)$ and $N' = (S', T', F', M'_0, \ell')$ be finitary nets, with N' unlabelled, $S' \subseteq S$, and $M'_0 = M_0 \upharpoonright S'$. Suppose:

- (a) $\forall t \in T, \ell(t) \neq \tau. \exists t' \in T', \ell(t') = \ell(t). \exists G \in \mathbb{N}^T, \ell(G) \equiv \emptyset. \llbracket t' \rrbracket = \llbracket t + G \rrbracket$.
- (b) For any $G \in \mathbb{Z}^T$ with $\ell(G) \equiv \emptyset$, $M' \in \mathbb{N}^{S'}$, $U' \in \mathbb{N}^{T'}$ and $U \in \mathbb{N}^T$ with $\ell'(U') = \ell(U)$, $M' + \bullet U' \in [M'_0]_{N'}$ and $M := M' + \bullet U' + (M_0 - M'_0) + \llbracket G \rrbracket - \bullet U \in \mathbb{N}^S$ with $M + \bullet U \in [M_0]_N$, it holds that:
 - (a) there is no infinite sequence $M \xrightarrow{\tau} M_1 \xrightarrow{\tau} M_2 \xrightarrow{\tau} \dots$
 - (b) if $M' \xrightarrow{a}$ with $a \in \text{Act}$ then $M \xrightarrow{a}$ or $M \xrightarrow{\tau}$
 - (c) and if $M \xrightarrow{a}$ with $a \in \text{Act}$ then $M' \xrightarrow{a}$.

Then $N \approx_{bSTb}^{\Delta} N'$.

Proof:³ Define $\mathcal{B} \subseteq (\mathbb{N}^S \times \mathbb{N}^T) \times (\mathbb{N}^{S'} \times \mathbb{N}^{T'})$ by $(M, U)\mathcal{B}(M', U') : \Leftrightarrow \ell'(U') = \ell(U) \wedge M' + \bullet U' \in [M'_0]_{N'} \wedge \exists G \in \mathbb{Z}^T. \ell(G) \equiv \emptyset \wedge M + \bullet U = M' + \bullet U' + (M_0 - M'_0) + \llbracket G \rrbracket \in [M_0]_N$. It suffices to show that \mathcal{B} satisfies Conditions (a)–(e) of Lemma 14.

(a) Take $G = \emptyset$.

(b) Suppose $(M_1, U_1)\mathcal{B}(M'_1, U'_1)$ and $(M_1, U_1) \xrightarrow{\tau} (M_2, U_2)$. Then $\ell'(U'_1) = \ell(U_1) \wedge M'_1 + \bullet U'_1 \in [M'_0]_{N'} \wedge \exists G \in \mathbb{Z}^T. \ell(G) \equiv \emptyset \wedge M_1 = M'_1 + \bullet U'_1 + (M_0 - M'_0) + \llbracket G \rrbracket - \bullet U_1 \wedge M_1 + \bullet U \in [M_0]_N$ and moreover $M_1 \xrightarrow{\tau} M_2 \wedge U_2 = U_1$. So $M_1[t]M_2$ for some $t \in T$ with $\ell(t) = \tau$. Hence $M_2 = M_1 + \llbracket t \rrbracket = M'_1 + \bullet U'_1 + (M_0 - M'_0) + \llbracket G + t \rrbracket - \bullet U_1$. Since $(M_1 + \bullet U_1)[t](M_2 + \bullet U_1)$, we have $M_2 + \bullet U_1 \in [M_0]_N$. Since also $\ell(G + t) \equiv \emptyset$ it follows that $(M_2, U_1)\mathcal{B}(M'_1, U'_1)$.

(c) Suppose $(M_1, U_1)\mathcal{B}(M'_1, U'_1)$ and $(M_1, U_1) \xrightarrow{\eta} (M_2, U_2)$, with $\eta \in \text{Act}^\pm$. Then $\ell'(U'_1) = \ell(U_1)$, $M'_1 + \bullet U'_1 \in [M'_0]_{N'}$ and

$$\exists G \in \mathbb{Z}^T. \ell(G) \equiv \emptyset \wedge M_1 + \bullet U_1 = M'_1 + \bullet U'_1 + (M_0 - M'_0) + \llbracket G \rrbracket \in [M_0]_N. \quad (5.1)$$

First suppose $\eta = a^+$. Then $\exists t \in T. \ell(t) = a \wedge M_1[t]M_2 = M_1 - \bullet t \wedge U_2 = U_1 + \{t\}$. Using that $M_1 \xrightarrow{a}$ with $a \in \text{Act}$, by Condition bc we have $M'_1 \xrightarrow{a}$, i. e. $M'_1[t']$ for some $t' \in T$ with $\ell'(t') = a$. Let $M'_2 := M'_1 - \bullet t$ and $U'_2 := U'_1 + \{t'\}$. Then $(M'_1, U'_1) \xrightarrow{a^+} (M'_2, U'_2)$. Moreover, $\ell(U_2) = \ell(U'_2)$, $M'_2 + \bullet U'_2 = M'_1 + \bullet U'_1 \in [M'_0]_{N'}$ and $M_2 + \bullet U_2 = M_1 + \bullet U_1$. In combination with (5.1) this yields

$$\begin{aligned} M_2 + \bullet U_2 = M_1 + \bullet U_1 &= M'_1 + \bullet U'_1 + (M_0 - M'_0) + \llbracket G \rrbracket \\ &= M'_2 + \bullet U'_2 + (M_0 - M'_0) + \llbracket G \rrbracket, \end{aligned}$$

so $(M_2, U_2)\mathcal{B}(M'_2, U'_2)$.

Now suppose $\eta = a^-$. Then $\exists t \in U_1. \ell(t) = a \wedge U_2 = U_1 - \{t\} \wedge M_2 = M_1 + t^\bullet$. Since $\ell'(U'_1) = \ell(U_1)$ there is a $t' \in U'_1$ with $\ell'(t') = a$. Let $M'_2 := M'_1 + t'^\bullet$ and $U'_2 := U'_1 - \{t'\}$. Then $(M'_1, U'_1) \xrightarrow{a^-} (M'_2, U'_2)$. By construction, $\ell(U_2) = \ell(U'_2)$. Moreover, $M_2 + \bullet U_2 = M_1 + t^\bullet + \bullet U_1 - \bullet t = (M_1 + \bullet U_1) + \llbracket t \rrbracket$, and likewise

$$M'_2 + \bullet U'_2 = (M'_1 + \bullet U'_1) + \llbracket t' \rrbracket \quad (5.2)$$

so $(M'_1 + \bullet U'_1)[t'](M'_2 + \bullet U'_2)$. Since $M'_1 + \bullet U'_1 \in [M'_0]_{N'}$, this yields $M'_2 + \bullet U'_2 \in [M'_0]_{N'}$. Moreover, $M_2 + \bullet U_2 = M_1 + t^\bullet + \bullet U_1 - \bullet t = M_1 + \bullet U_1 + \llbracket t \rrbracket \in [M_0]_N$. Furthermore, combining (5.1) and (5.2) gives

$$\exists G \in \mathbb{Z}^T. \ell(G) \equiv \emptyset \wedge M_2 + \bullet U_2 - \llbracket t \rrbracket = M'_2 + \bullet U'_2 - \llbracket t' \rrbracket + (M_0 - M'_0) + \llbracket G \rrbracket. \quad (5.3)$$

By Condition a of Lemma 15, $\exists t'' \in T', \ell(t'') = \ell(t)$. $\exists G_t \in \mathbb{N}^T, \ell(G_t) \equiv \emptyset. \llbracket t \rrbracket = \llbracket t'' - G_t \rrbracket$. Since N' is an unlabelled net, it has only one transition t^\dagger with $\ell(t^\dagger) = a$, so $t'' = t'$. Substitution of $\llbracket t' - G_t \rrbracket$ for $\llbracket t \rrbracket$ in (5.3) yields

$$\exists G \in \mathbb{Z}^T. \ell(G) \equiv \emptyset \wedge M_2 + \bullet U_2 = M'_2 + \bullet U'_2 + (M_0 - M'_0) + \llbracket G - G_t \rrbracket.$$

³For didactic reason it may be preferable to skip ahead and read the (simpler) proof of Lemma 18 first.

Since $\ell(G - G_t) \equiv \emptyset$ we obtain $(M_2, U_2)\mathcal{B}(M'_2, U'_2)$.

(d) Follows directly from Condition [bb](#) and Definition [16](#).

(e) Follows directly from Condition [ba](#) and Definition [16](#). \square

To illustrate the use of Lemmas [18](#) and [15](#), let N' be an unlabelled net and N be its conflict replicating implementation, depicted in Figure [5.5](#). Condition (1) says that for any visible transition t in the implementation – this must be execute_j^i for some i and j – there must be a transition t' in N' with the same label – this must be i – such that the same token replacement $\llbracket t' \rrbracket$ that results from firing t' in the net N' can also be achieved by t in N together with a multiset G of internal transitions of N . For this to even make sense it is necessary that $S' \subseteq S$, so that $\llbracket t' \rrbracket$ can just as well be seen as a token replacement of N . This condition can be fulfilled by taking G to contain distribute_p for every preplace p of i , $\text{fetch}_{i,j}^{p,c}$ for every preplace p of i and every $c \in p^\bullet$, fetched_j^i , $u \cdot \text{elide}_i$ for $u \in \Omega_i$, and finalise_i .

In the proof of Lemma [18/15](#), a branching bisimulation is constructed between the markings of N' and N , by relating any reachable marking M' of N' with the corresponding marking $M' + (M_0 - M'_0)$ of N ; the latter is the marking M' seen as a marking of N , together with those places in $S \setminus S'$ that are marked initially (or by default). In addition, M' is also related to markings obtained from $M' + (M_0 - M'_0)$ by adding or subtracting the token replacement due to firing some internal transitions of N . For instance, compared to the state of N given by the marking $M' + (M_0 - M'_0)$ it could be that finalise_i has not yet fired – so that $\text{ack}_i(t)$ is marked for all $t \in \Omega_i$ instead of the postplaces r of i – and that distribute_p has already fired for some place p . This gives rise to the marking $M' + (M_0 - M'_0) + \llbracket G \rrbracket$ being related to M' , with $G = -\{\text{finalise}_i\} + \{\text{distribute}_p\}$. To show that the relation really is a branching bisimulation with explicit divergence it suffices to check the conditions (a)–(c). That these are enough to obtain the stronger conditions (a)–(e) of Lemma [17/14](#) follows with help of the new condition (1).

In the proof of Lemma [15](#) the bisimulation constructed in the proof of Lemma [18](#) is strengthened to a split bisimulation by taking account of the sets U' and U of transitions currently firing in N' and N . Here we need to require that U' and U carry the same multiset of labels. Moreover, the preplaces of U' and U need to be added to M' and M when determining that they are reachable markings, and in relating these markings to each other; for these purposes we thus use the markings we would have had before starting the transitions that are currently firing. On the other hand, M' and M themselves need to be markings (i.e. put a non-negative number of tokens in each place), and in conditions (a)–(c) only those transitions matter that can be fired from M' and M themselves – without the preplaces of U' and U .

In Lemma [15](#) a relation is explored between markings \bar{M} and $\bar{M} + \llbracket H \rrbracket$ (where \bar{M} is $M' + \bullet U' + (M_0 - M'_0)$ of Lemma [15](#), $H := G$, and $\bar{M} + \llbracket H \rrbracket$ is $M + \bullet U$ of Lemma [15](#)). In such a case, we can think of \bar{M} as an “original marking”, and of $\bar{M} + \llbracket H \rrbracket$ as a modification of this marking by the token replacement $\llbracket H \rrbracket$. The

next lemma provides a method to trace certain places s marked by $\bar{M} + \llbracket H \rrbracket$ (or transitions t that are enabled under $\bar{M} + \llbracket H \rrbracket$) back to places that must have been marked by \bar{M} before taking into account the token replacement $\llbracket H \rrbracket$. Such places are called *faithful origins* of s (or t). In tracking the faithful origins of places and transitions, we assume that the places marked by \bar{M} are taken from a set S_+ and the transitions in H from a set T_+ . In Lemma 16 we furthermore assume that the flow relation restricted to $S \cup T_+$ is acyclic. We will need this lemma in proving the correctness of our final method of proving $N \approx_{bSTb}^\Delta N'$.

Definition 49 Let $N = (S, T, F, M_0, \ell)$ be a finitary net, $T_+ \subseteq T$ a set of transitions and $S_+ \subseteq S$ a set of places.

- A *path* in N is an alternating sequence $\pi = x_0 x_1 x_2 \cdots x_n \in (S \cup T)^*$ of places and transitions, such that $F(x_i, x_{i+1}) > 0$ for $0 \leq i < n$. The *arc weight* $F(\pi)$ of such a path is the product $\prod_0^{n-1} F(x_i, x_{i+1})$.
- A place $s \in S$ is *faithful* w.r.t. T_+ and S_+ iff $|\{s\} \cap S_+| + \sum_{t \in T_+} F(t, s) = 1$.
- A path $x_0 x_1 x_2 \cdots x_n \in (S \cup T)^*$ from x_0 to x_n is *faithful* w.r.t. T_+ and S_+ iff all intermediate nodes x_i for $0 \leq i < n$ are either transitions in T_+ or faithful places w.r.t. T_+ and S_+ .
- For $x \in S \cup T$, the *infinitary multiset* $*x \in (\mathbb{N} \cup \{\infty\})^{S_+}$ of *faithful origins* of x is given by $*x(s) = \sup\{F(\pi) \mid \pi \text{ is a faithful path from } s \in S_+ \text{ to } x\}$. (So $*x(s) = 0$ if no such path exists.)

Suppose a marking M is reachable from a marking $\bar{M} \in \mathbb{N}^{S_+}$ by firing transitions from T_+ only. So $M = \bar{M} + \llbracket H \rrbracket$ for some $H \in \mathbb{N}^{T_+}$. Then, if a faithful place s bears a token under M – i.e. $M(s) > 0$ – this token has a unique source: if $s \in S_+$ it must stem from \bar{M} and otherwise it must be produced by the unique transition $t \in T_+$ with $F(t, s) = 1$.

Now consider a period in the evolution of the net N that starts with the marking \bar{M} , and during which only transitions from T_+ fire. Suppose that $\pi = x_0 x_1 x_2 \cdots x_n$ is a faithful path from a place $x_0 \in S_+$ to a either a faithful place x_n that gets marked at some point during this period or a transition x_n that fires during (or right after) this period. In that case a token, left on x_0 by the marking \bar{M} , must have travelled along that path from x_0 to x_n – where a token is understood to visit a transition when that transition fires. Namely, if x_{i+1} is a transition that fired at some point, then its (faithful) preplace x_i must have been marked right beforehand; and if a faithful place x_{j+i} was marked at some point, then $x_{j+i} \notin S_+$ and the token in x_{j+i} must have been produced by the transition $x_i \in T_+$.

Note that $F(\pi)$ is the product of all arc weights in the path on arcs from places to transitions; for all the weights on arcs from transitions in T_+ to faithful places are 1. Taking arc weights into account, for every token in x_n as many as $F(\pi)$ token must have started in x_0 . Namely, for a transition x_{i+1} to fire once, $F(x_i, x_{i+1})$ tokens must have come from place x_i , and for each token in a faithful place x_{j+1} , the transition x_j must have fired once.

In a net without arc weights, $*x$ is always a set, namely the set of places s in S_+ from which the flow relation of the net admits a path to x that passes only through faithful places and transitions from T_+ (with the possible exception of x itself). For nets with arc weights, the underlying set of $*x$ is the same, and the multiplicity of $s \in *x$ is obtained by multiplying all arc weights on the qualifying path from s to x ; in case of multiple such paths, we take the upper bound over all such paths (which could yield the value ∞). It follows from the analysis above that if a faithful place x gets marked, or a transition x enabled, during a period as described above, then at least $*x(s)$ tokens must have been present in s at the beginning of this period. Lemma 16 formalises this analysis by comparing a marking $\bar{M} + \llbracket H \rrbracket$ that marks or enables x (possibly multiple times) with the marking \bar{M} that marks the faithful origins $*x$ of x . Here $H \in \mathbb{N}^{T_+}$ is the multiset of transitions whose firing converts \bar{M} into $\bar{M} + \llbracket H \rrbracket$. However, Lemma 16 does not require that this multiset actually can be fired in any particular order. To enable that generalisation, it must assume that $F \upharpoonright (S \cup T_+)$ is acyclic.

For $k \neq 0$, we have

$$k \cdot *x(s) = \sup\{k \cdot F(\pi) \mid \pi \text{ is a faithful path from } s \in S_+ \text{ to } x\}.$$

In order to also have this equality for $k = 0$ and $*x(s) = \infty$ we define $0 \cdot \infty := 0$ in this context.

Observation 12 Let (S, T, F, M_0, ℓ) be a finitary net, $T_+ \subseteq T$ a set of transitions and $S_+ \subseteq S$ a set of places. For faithful places s and transitions $t \in T$ we have

$$*s = \begin{cases} \{s\} & \text{if } s \in S_+ \\ *t & \text{if } t \in T_+ \wedge F(t, s) = 1 \end{cases} \quad *t = \bigcup \{F(s, t) \cdot *s \mid s \in \bullet t \wedge s \text{ faithful}\}.$$

□

Lemma 16 Let (S, T, F, M_0, ℓ) be a finitary net, $T_+ \subseteq T$ a set of transitions such that $F \upharpoonright (S \cup T_+)$ is acyclic, and $S_+ \subseteq S$ a set of places. Let $\bar{M} \in \mathbb{N}^{S_+}$ and $H \in \mathbb{N}^{T_+}$, such that $\bar{M} + \llbracket H \rrbracket \in \mathbb{N}^S$ (i. e. places occur only non-negatively in $\bar{M} + \llbracket H \rrbracket$). Then

- (a) for any faithful place s w.r.t. T_+ and S_+ we have $(\bar{M} + \llbracket H \rrbracket)(s) \cdot *s \leq \bar{M}$;
- (b) for any $k \in \mathbb{N}$, and any transition t with $(\bar{M} + \llbracket H \rrbracket)[k \cdot \{t\}]$, we have $k \cdot *t \leq \bar{M}$.

Proof: We apply induction on $|H|$. In the base case, $H = \emptyset$, which formally is included in the induction step, (a) follows directly from the assumption that $\bar{M} \in \mathbb{N}^{S_+}$ and the observation that $*s = \{s\}$.

(a). When $(\bar{M} + \llbracket H \rrbracket)(s) = 0$ it trivially follows that $(\bar{M} + \llbracket H \rrbracket)(s) \cdot *s \leq \bar{M}$. So suppose $(\bar{M} + \llbracket H \rrbracket)(s) > 0$. Then either $s \in S_+$ or there is a unique $t \in T_+$ with $H(t) > 0$ and $F(t, s) = 1$. In the first case, using that $s \in u^\bullet$ for no $u \in T_+$, we have $(\bar{M} + \llbracket H \rrbracket)(s) \leq \bar{M}(s)$, so $(\bar{M} + \llbracket H \rrbracket)(s) \cdot *s \leq \bar{M}(s) \cdot \{s\} \leq \bar{M}$. In the latter

case, we have $(\bar{M} + \llbracket H \rrbracket)(s) \leq \bar{M}(s) + \sum_{u \in T_+} H(u) \cdot F(u, s) = \bar{M}(s) + H(t) = H(t)$ and $*s = *t$. Thus:

$$(\bar{M} + \llbracket H \rrbracket)(s) \cdot *s \leq H(t) \cdot *t. \quad (5.4)$$

Let $U := \{u \in T_+ \mid H(u) > 0 \wedge uF^+t\}$ be the set of transitions occurring in H from which the flow relation of the net offers a non-empty path to t . As $F \upharpoonright (S \cup T_+)$ is acyclic, $t \notin U$, so $H \upharpoonright U < H$. Let s' be any place with $s' \in \bullet u$ for some transition $u \in U$. Then, by construction of U , it cannot happen that $s' \in v^\bullet$ for some transition $v \notin U$ with $H(v) > 0$. Hence $(\bar{M} + \llbracket H \upharpoonright U \rrbracket)(s') \geq (\bar{M} + \llbracket H \rrbracket)(s') \geq 0$. Moreover, for any other place s'' we have $\bullet(H \upharpoonright U)(s'') = 0$ and thus $(\bar{M} + \llbracket H \upharpoonright U \rrbracket)(s'') \geq \bar{M}(s'') \geq 0$. It follows that $\bar{M} + \llbracket H \upharpoonright U \rrbracket \in \mathbb{N}^S$.

For each $s''' \in \bullet t$ we have $\bullet(H - H \upharpoonright U)(s''') \geq H(t) \cdot \bullet t(s''')$ and $(H - H \upharpoonright U) \bullet(s''') = 0$ and hence $0 \leq (\bar{M} + \llbracket H \rrbracket)(s''') \leq (\bar{M} + \llbracket H \upharpoonright U \rrbracket)(s''') - H(t) \cdot \bullet t(s''')$. For this reason, $H(t) \cdot \bullet t \leq \bar{M} + \llbracket H \upharpoonright U \rrbracket$. It follows that $(\bar{M} + \llbracket H \upharpoonright U \rrbracket)[H(t) \cdot \{t\}]$. Thus, by (5.4) and induction, $(\bar{M} + \llbracket H \rrbracket)(s) \cdot *s \leq H(t) \cdot *t \leq \bar{M}$.

(b). Let $(\bar{M} + \llbracket H \rrbracket)[k \cdot \{t\}]$. For any faithful $s \in \bullet t$ we have $(\bar{M} + \llbracket H \rrbracket)(s) \geq k \cdot F(s, t)$, and thus, using (a),

$$k \cdot F(s, t) \cdot *s \leq (\bar{M} + \llbracket H \rrbracket)(s) \cdot *s \leq \bar{M}.$$

Therefore, by Observation 12, $k \cdot *t = \bigcup \{k \cdot F(s, t) \cdot *s \mid s \in \bullet t \wedge s \text{ faithful}\} \leq \bar{M}$. \square

As a (forthcoming) application of Lemma 16 consider the branching split bisimulation with explicit divergence between a net N' and its conflict replicating implementation N that is constructed according to the proof of Lemma 15. When a split marking (M', U') is related to (M, U) , then $M + \bullet U = M' + \bullet U' + (M_0 - M'_0) + \llbracket G \rrbracket$ for a signed multiset G of internal transitions of N . Furthermore suppose that G is a true multiset over the set of transitions T_+ , consisting of distribute_p , $\text{initialise}_j \cdot \text{fire}$ and $\text{transfer}_j^h \cdot \text{fire}$ only (for arbitrary p, j and h). Take $\bar{M} := M' + \bullet U' + (M_0 - M'_0)$, $H := G$ and thus $M + \bullet U = \bar{M} + \llbracket H \rrbracket$. Let $S_+ := S' \cup \{s \in S \mid (M_0 - M'_0)(s) > 0\}$. Then $p \text{ distribute}_p \ p_i \text{ initialise}_i \cdot \text{fire} \ \text{pre}_j^i \ \text{execute}_j^i$ is a faithful path from p to execute_j^i . The arc weight of this path is $F'(p, i)$. So $*\text{execute}_j^i \geq F'(p, i)$. Thus if execute_j^i is enabled under $M + \bullet U$ then \bar{M} must place at least $F'(p, i)$ tokens in the place p . As this reasoning applies to every preplace p of i , it follows that i is enabled under $M' + \bullet U'$.

The following theorem is the main result of this section. It presents a method for proving $N \approx_{bSTb}^\Delta N'$ for N a net and N' an unlabelled net. Its main advantage w.r.t. directly using the definition, or w.r.t. application of Lemma 14 or 15, is the replacement of requirements on the dynamic behaviour of nets by structural requirements. Such requirements are typically easier to check. Replacing the requirement “ $M + \bullet U \in [M_0]_N$ ” in Condition 5 by “ $M + \bullet U \in \mathbb{N}^S$ ” would have yielded an even more structural version of this theorem; however, that version turned out not to be strong enough for the verification task performed in Section 5.3.

Theorem 9 Let $N = (S, T, F, M_0, \ell)$ and $N' = (S', T', F', M'_0, \ell')$ be finitary nets with N' unlabelled, $S' \subseteq S$, and $M'_0 = M_0 \upharpoonright S'$. Suppose there exist sets $T_+ \subseteq T$ and $T_- \subseteq T$ and a class $NF \subseteq \mathbb{Z}^T$, such that

1. $F \upharpoonright (S \cup T_+)$ is acyclic.
2. $F \upharpoonright (S \cup T_-)$ is acyclic.
3. For all $t \in T$ with $\ell(t) \neq \tau$ there exists some $t' \in T'$ with $\ell(t') = \ell(t)$ such that $\bullet t' \leq \bullet t \wedge \exists G \in \mathbb{N}^T, \ell(G) \equiv \emptyset, \llbracket t' \rrbracket = \llbracket t + G \rrbracket$. Here $\bullet t$ is the multiset of faithful origins of t w.r.t. T_+ and $S' \cup \{s \in S \mid M_0(s) > 0\}$.
4. There exists a function $f : T \rightarrow \mathbb{N}$ with $f(t) > 0$ for all $t \in T$, extended to \mathbb{Z}^T as in Definition 1, such that for each $G \in \mathbb{Z}^T$ with $\ell(G) \equiv \emptyset$ there is an $H \in NF$ with $\ell(H) \equiv \emptyset, \llbracket H \rrbracket = \llbracket G \rrbracket$ and $f(H) = f(G)$.
5. For every $M' \in \mathbb{N}^{S'}, U' \in \mathbb{N}^{T'}$ and $U \in \mathbb{N}^T$ with $\ell(U) = \ell'(U')$ and $M' + \bullet U' \in [M'_0]_{N'}$, there is an $H_{M',U} \in \mathbb{N}^{T_+}$ with $\ell(H_{M',U}) \equiv \emptyset$, such that for each $H \in NF$ with $M := M' + \bullet U' + (M_0 - M'_0) + \llbracket H \rrbracket - \bullet U \in \mathbb{N}^S$ and $M + \bullet U \in [M_0]_N$:
 - (a) $M_{M',U} := M' + \bullet U' + (M_0 - M'_0) + \llbracket H_{M',U} \rrbracket - \bullet U \in \mathbb{N}^S$,
 - (b) if $M' \xrightarrow{a}$ with $a \in \text{Act}$ then $M_{M',U} \xrightarrow{a}$,
 - (c) $H \leq H_{M',U}$.
 - (d) if $H(u) < 0$ then $u \in T_-$,
 - (e) if $H(u) < 0$ and $H(t) > 0$ then $\bullet u \cap \bullet t = \emptyset$,
 - (f) if $H(u) < 0$ and $(M + \bullet U)[t]$ with $\ell(t) \neq \tau$ then $\bullet u \cap \bullet t = \emptyset$,
 - (g) if $(M + \bullet U)[\{t\} + \{u\}]$ and $t', u' \in T'$ with $\ell'(t') = \ell(t)$ and $\ell'(u') = \ell(u)$, then $\bullet t' \cap \bullet u' = \emptyset$.

Then $N \approx_{bSTb}^{\Delta} N'$.

Proof: It suffices to show that Condition b of Lemma 15 holds (as Condition a of Lemma 15 is part of Condition 3 above). So let $G \in \mathbb{Z}^T$ with $\ell(G) \equiv \emptyset$, $M' \in \mathbb{N}^{S'}, U' \in \mathbb{N}^{T'}$ and let $U \in \mathbb{N}^T$ with $\ell'(U') = \ell(U)$, $M' + \bullet U' \in [M'_0]_{N'}$, $M := M' + \bullet U' + (M_0 - M'_0) + \llbracket G \rrbracket - \bullet U \in \mathbb{N}^S$ and $M + \bullet U \in [M_0]_N$.

- (a) Suppose $M \xrightarrow{\tau} M_1 \xrightarrow{\tau} M_2 \xrightarrow{\tau} \dots$. Then there are transitions $t_i \in T$ with $\ell(t_i) = \tau$, for all $i \geq 1$, such that $M[t_1]M_1[t_2]M_2[t_3]\dots$. As also $(M + \bullet U)[t_1](M_1 + \bullet U)[t_2](M_2 + \bullet U)[t_3]\dots$, it follows that $(M_i + \bullet U) \in [M_0]_N$ for all $i \geq 1$. Let $G_0 := G$ and for all $i \geq 1$ let $G_i := G_{i-1} + \{t_i\}$. Then $\ell(G_i) \equiv \emptyset$ and $M_i = M' + \bullet U' + (M_0 - M'_0) + \llbracket G_i \rrbracket - \bullet U$. Moreover, $f(G_i) = f(G_{i-1}) + f(t_i) > f(G_{i-1})$. For all $i \geq 0$, using Condition 4, let $H_i \in NF$ be so that $\llbracket H_i \rrbracket = \llbracket G_i \rrbracket$ and $f(H_i) = f(G_i)$. Then $M_i = M' + \bullet U' + (M_0 - M'_0) + \llbracket H_i \rrbracket - \bullet U$ and $f(H_0) < f(H_1) < f(H_2) < \dots$. However, from Condition 5c we get $f(H_i) \leq f(H_{M'})$ for all $i \geq 0$. The sequence $M \xrightarrow{\tau} M_1 \xrightarrow{\tau} M_2 \xrightarrow{\tau} \dots$ therefore must be finite.

- (b) Now suppose $M' \xrightarrow{a}$ with $a \in \text{Act}$. By Condition 4 above there exists an $H \in NF$ such that $\ell(H) \equiv \emptyset$ and $\llbracket H \rrbracket = \llbracket G \rrbracket$, and hence $M = M' + \bullet U' + (M_0 - M'_0) + \llbracket H \rrbracket - \bullet U$. Let $H_- := \{u \in T \mid H(u) < 0\}$.

- First suppose $H_- \neq \emptyset$. By Condition 5d, $H_- \subseteq T_-$. By Condition 2, the relation $<_- := (F \upharpoonright (S \cup T_-))^+$ is a partial order on $S \cup T_-$, and hence on H_- . Let u be a minimal transition in H_- w.r.t. $<_-$. By definition, for all $s \in S$,

$$\begin{aligned} M(s) = & M'(s) + \bullet U'(s) + (M_0 - M'_0)(s) + \\ & \sum_{t \in T} H(t) \cdot F(t, s) + \sum_{t \in T^-} -H(t) \cdot F(s, t) + \\ & \sum_{t \in U} -U(t) \cdot F(t, s). \end{aligned} \quad (5.5)$$

As $M'_0 = M_0 \upharpoonright S'$, we have $M'_0 \leq M_0$. Hence the first three summands in this equation are always non-negative. Now assume $s \in \bullet u$. Since u is minimal w.r.t. $<_-$, there is no $t \in T$ with $H(t) < 0$ and $F(t, s) \neq 0$. Hence also all summands $H(t) \cdot F(t, s)$ are non-negative. By Condition 5e, there is no $t \in T$ with $H(t) > 0$ and $F(s, t) \neq 0$, so all summands $-H(t) \cdot F(s, t)$ are non-negative as well. By Condition 5f, there is no $t \in T$ with $U(t) > 0$ and $F(s, t) \neq 0$, for this would imply that $\ell(t) \neq \tau$ and $(M + \bullet U)[t]$, so no summands in (5.5) are negative. Thus $0 \leq -H(u) \cdot F(s, u) \leq M(s)$. Since $H(u) \leq -1$, this implies $M(s) \geq F(s, u)$. Hence u is enabled in M . As $\ell(u) = \tau$, we have $M \xrightarrow{\tau}$.

- Next suppose $H_- = \emptyset$ but $H \neq H_{M', U}$. Let $H^\sim := \{u \in T \mid H_{M', U}(u) - H(u) > 0\}$. Then $H^\sim \neq \emptyset$ by Condition 5c. Since $H_{M', U} \in \mathbb{N}^{T_+}$, $H^\sim \subseteq T_+$. By Condition 1, $<_+ := (F \upharpoonright (S \cup T_+))^+$ is a partial order on $S \cup T_+$, and hence on H^\sim . Let u be a minimal transition in H^\sim w.r.t. $<_+$. We have $M = M' + \bullet U' + (M_0 - M'_0) + \llbracket H_{M', U} + (H - H_{M', U}) \rrbracket - \bullet U = M_{M', U} + \llbracket H - H_{M', U} \rrbracket$. Hence, for all $s \in S$,

$$\begin{aligned} M(s) = & M_{M', U}(s) + \sum_{t \in T} (H - H_{M', U})(t) \cdot F(t, s) + \\ & \sum_{t \in T^-} -(H - H_{M', U})(t) \cdot F(s, t). \end{aligned} \quad (5.6)$$

By Condition 5a, $M_{M', U} \in \mathbb{N}^S$. By Condition 5c, $H - H_{M', U} \leq \emptyset$. For $s \in \bullet u$ there is moreover no $t \in H^\sim$ with $s \in t^\bullet$, so no $t \in T$ with $(H - H_{M', U})(t) < 0$ and $F(t, s) \neq 0$. Hence no summands in (5.6) are negative. It thereby follows that $0 \leq -(H - H_{M', U})(u) \cdot F(s, u) \leq M(s)$. Since $(H - H_{M', U})(u) \leq -1$, this implies $M(s) \geq F(s, u)$. Hence u is enabled in M . As $\ell(u) = \tau$, we have $M \xrightarrow{\tau}$.

- Finally suppose $H = H_{M', U}$. Then $M = M_{M', U}$ and $M \xrightarrow{a}$ follows by Condition 5b.

- (c) Next suppose $M \xrightarrow{a}$ with $a \in \text{Act}$. Then there is a $t \in T$ with $\ell(t) = a \neq \tau$ and $M[t]$. So $(M + \bullet U)[t]$. We will first show that $(M' + \bullet U') \xrightarrow{a}$. By Condition 4 there exists an $H_0 \in NF \subseteq \mathbb{Z}^T$ such that $\ell(H_0) \equiv \emptyset$ and

$\llbracket H_0 \rrbracket = \llbracket G \rrbracket$, and hence $M + \bullet U = M' + \bullet U' + (M_0 - M'_0) + \llbracket H_0 \rrbracket \in [M_0]_N$. For our first step, it suffices to show that whenever $H \in NF$ with $M_H := M' + \bullet U' + (M_0 - M'_0) + \llbracket H \rrbracket \in [M_0]$ and $M_H[t]$, then $(M' + \bullet U') \xrightarrow{a}$. We show this by induction on $f(H_{M',U} - H)$, observing that $f(H_{M',U} - H) \in \mathbb{N}$ by Conditions 5c (with empty U) and 4.

We consider two cases, depending on the emptiness of $H_- := \{u \in T \mid H(u) < 0\}$.

First assume $H_- = \emptyset$. Then $H \in \mathbb{N}^T$. By Condition 5c (with empty U) we even have $H \in \mathbb{N}^{T_+}$. Let $*t$ denote the multiset of faithful origins of t w.r.t. T_+ and $S_+ := S' \cup \{s \in S \mid M_0(s) > 0\}$. By Lemma 16(b), taking $k = 1$ and $\bar{M} := M' + \bullet U' + (M_0 - M'_0)$, and using Condition 1 of Theorem 9, $*t \leq M' + \bullet U' + (M_0 - M'_0)$. So by Condition 3 of Theorem 9 there is a $t' \in T'$ with $\ell(t') = \ell(t)$ and $\bullet t' \leq M' + \bullet U' + (M_0 - M'_0)$. Since $\bullet t' \in \mathbb{N}^{S'}$ and $M'_0 = M_0 \upharpoonright S'$, this implies $\bullet t' \leq M' + \bullet U'$. It follows that $(M' + \bullet U')[t']_N$ and hence $(M' + \bullet U') \xrightarrow{a}$.

Now assume $H_- \neq \emptyset$. By the same proof as for (b) above, case $H_- \neq \emptyset$, there is a transition $u \in H_-$ that is enabled in M_H . So $M_H[u]M_1$ for some $M_1 \in [M_0]_N$, and $M_1 = M' + \bullet U' + (M_0 - M'_0) + \llbracket H + u \rrbracket$. By Condition 5f of Theorem 9 (still with empty U), $\bullet u \cap \bullet t = \emptyset$, and thus $M_1[t]$. By Condition 4 of Theorem 9 there exists an $H_1 \in NF$ such that $\ell(H_1) \equiv \emptyset$, $\llbracket H_1 \rrbracket = \llbracket H + u \rrbracket$, and $f(H_1) = f(H + u) > f(H)$. Thus $M_1 = M_{H_1}$ and $f(H_{M',U} - H_1) < f(H_{M',U} - H)$. By induction we obtain $(M' + \bullet U') \xrightarrow{a}$.

By the above reasoning, there is a $t' \in T'$ such that $\ell'(t') = \ell(t)$ and $(M' + \bullet U')[t']$. Now take any $u' \in U'$. Then there must be an $u \in U$ with $\ell'(u') = \ell(u)$. Since $M[t]$, we have $(M + \bullet U)[\{t\} + \{u\}]$ and by Condition 5g we obtain $\bullet t' \cap \bullet u' = \emptyset$. It follows that $M'[t']$, and hence $M' \xrightarrow{a}$. \square

Theorem 9 will be applied in Section 5.3 to show the correctness of our conflict replicating implementation N of a given net N' . A crucial observation about N is that its internal transitions can be partitioned into a set T_+ of transitions (3 boxes in Figure 5.5) that have to occur before firing execute_j^i (for some i and j) and a set T_- of transitions (14 boxes) that can only occur afterwards. In the construction of our bisimulation we consider markings of the form $M' + \bullet U' + (M_0 - M'_0) + \llbracket H \rrbracket$, where H is a signed multiset of internal transitions that tells how much the marking deviates from the marking $M' + \bullet U' + (M_0 - M'_0)$ of N . The bisimulation relates both markings of N to the marking $M' + \bullet U'$ of N' . When an internal transition of N fires, the related marking of N' remains the same. However, when N fires a visible transition execute_j^i then the related marking of N' becomes $M' + \bullet U' + \llbracket i \rrbracket$, so in view of the structural property in Lemma 15(1), a new set H' can be calculated as $H' := H - G$, where G is the signed multiset for which $\llbracket i \rrbracket = \llbracket \text{execute}_j^i + G \rrbracket$. A consequence of this is that elements of T_+ only occur with positive multiplicities in H , whereas elements of T_- occur only with negative multiplicities.

To be precise, it may be that two different sets H_1 and H_2 yield the same token replacement, i. e. $\llbracket H_1 \rrbracket = \llbracket H_2 \rrbracket$. As a result of this, there may be multiple

ways to write a marking as $M' + \bullet U' + (M_0 - M'_0) + \llbracket H \rrbracket$ for given M' and U' . The above applies only when converting the signed multisets H to a normal form NF that eliminates this ambiguity.

For given M' and U' , the multiset $H_{M',U'}$ is an upper bound of the possible choices of H for which $M' + \bullet U' + (M_0 - M'_0) + \llbracket H \rrbracket$ can be a reachable marking. This is expressed by Condition 5c. If all internal transitions in $H_{M',U'}$ have fired, the next transition must be an external one. Now the conditions of Theorem 9 guarantee that as long as this upper bound is not reached, the net N can perform internal actions, and when it is reached (and possibly also beforehand) it can perform the same actions as the net N' under marking M' . Condition 4 moreover guarantees that this upper bound will be reached in finitely many steps. Due to the need to renormalise the signed multisets H after adding elements to them, this is not straightforward.

These considerations imply that transitions fired by N' can be simulated by N . The other direction involves similar arguments, together with an application of Lemma 16.

Digression: Interleaving semantics

Above, a method is presented for establishing the equivalence of two nets, one of which known to be unlabelled, up to branching ST-bisimilarity with explicit divergence. Here, we simplify this result into a method for establishing the equivalence of the two nets up interleaving branching bisimilarity with explicit divergence. This result might be of independent interest, but is not used in the following.

Lemma 17 Let $N = (S, T, F, M_0, \ell)$ and $N' = (S', T', F', M'_0, \ell')$ be two finitary nets, N' being unlabelled. Suppose there is a relation $\mathcal{B} \subseteq \mathbb{N}^S \times \mathbb{N}^{S'}$ such that

- (a) $M_0 \mathcal{B} M'_0$,
- (b) if $M_1 \mathcal{B} M'_1$ and $M_1 \xrightarrow{\tau} M_2$ then $M_2 \mathcal{B} M'_1$,
- (c) if $M_1 \mathcal{B} M'_1$ and $M_1 \xrightarrow{a} M_2$ for some $a \in \text{Act}$ then $\exists M'_2. M'_1 \xrightarrow{a} M'_2 \wedge M_2 \mathcal{B} M'_2$,
- (d) if $M_1 \mathcal{B} M'_1$ and $M'_1 \xrightarrow{a}$ for some $a \in \text{Act}$ then either $M_1 \xrightarrow{a}$ or $M_1 \xrightarrow{\tau}$
- (e) and there is no infinite sequence $M \xrightarrow{\tau} M_1 \xrightarrow{\tau} M_2 \xrightarrow{\tau} \dots$ with $M \mathcal{B} M'$ for some M' .

Then N and N' are interleaving branching bisimilar with explicit divergence.

Proof: Analogous to the proof of Lemma 14. □

Lemma 18 Let $N = (S, T, F, M_0, \ell)$ and $N' = (S', T', F', M'_0, \ell')$ be two finitary nets, N' being unlabelled, with $S' \subseteq S$ and $M'_0 = M_0 \upharpoonright S'$. Suppose:

1. $\forall t \in T, \ell(t) \neq \tau. \exists t' \in T', \ell(t') = \ell(t). \exists G \in \mathbb{N}^T, \ell(G) \equiv \emptyset. \llbracket t' \rrbracket = \llbracket t + G \rrbracket.$
2. For any $G \in \mathbb{Z}^T$ with $\ell(G) \equiv \emptyset, M' \in [M'_0]_{N'}$ and $M := M' + (M_0 - M'_0) + \llbracket G \rrbracket \in [M_0]_N$, it holds that:
 - (a) there is no infinite sequence $M \xrightarrow{\tau} M_1 \xrightarrow{\tau} M_2 \xrightarrow{\tau} \dots,$
 - (b) if $M' \xrightarrow{a}$ with $a \in \text{Act}$ then $M \xrightarrow{a}$ or $M \xrightarrow{\tau}$
 - (c) and if $M \xrightarrow{a}$ with $a \in \text{Act}$ then $M' \xrightarrow{a}$.

Then N and N' are interleaving branching bisimilar with explicit divergence.

Proof: Define $\mathcal{B} \subseteq \mathbb{N}^S \times \mathbb{N}^{S'}$ by

$$M \mathcal{B} M' :\Leftrightarrow M' \in [M'_0]_{N'} \wedge \exists G \in \mathbb{Z}^T. M = M' + (M_0 - M'_0) + \llbracket G \rrbracket \in [M_0]_N \wedge \ell(G) \equiv \emptyset.$$

It suffices to show that \mathcal{B} satisfies Conditions (a)–(e) of Lemma 17.

- (a) Take $G = \emptyset$.
- (b) Suppose $M_1 \mathcal{B} M'_1$ and $M_1 \xrightarrow{\tau} M_2$. Then $\exists G \in \mathbb{Z}^T. M_1 = M'_1 + (M_0 - M'_0) + \llbracket G \rrbracket \wedge \ell(G) \equiv \emptyset$ and $\exists t \in T. \ell(t) = \tau \wedge M_2 = M_1 + \llbracket t \rrbracket = M'_1 + (M_0 - M'_0) + \llbracket G + t \rrbracket$. Moreover, $M_1 \in [M_0]_N$ and hence $M_2 \in [M_0]_N$. Furthermore, $M'_1 \in [M'_0]_{N'}$ and $\ell(G + t) \equiv \emptyset$, so $M_2 \mathcal{B} M'_1$.
- (c) Suppose $M_1 \mathcal{B} M'_1$ and $M_1 \xrightarrow{a} M_2$. Then $\exists G \in \mathbb{Z}^T. M_1 = M'_1 + (M_0 - M'_0) + \llbracket G \rrbracket \wedge \ell(G) \equiv \emptyset$ and $\exists t \in T. \ell(t) = a \neq \tau \wedge M_2 = M_1 + \llbracket t \rrbracket = M'_1 + (M_0 - M'_0) + \llbracket G + t \rrbracket$. Moreover, $M_1 \in [M_0]_N$ and hence $M_2 \in [M_0]_N$. Furthermore, $M'_1 \in [M'_0]_{N'}$. By Condition 1 of Lemma 18, $\exists t' \in T', \ell(t') = \ell(t). \exists G_t \in \mathbb{N}^T, \ell(G_t) \equiv \emptyset. \llbracket t \rrbracket = \llbracket t' - G_t \rrbracket$. Substitution of $\llbracket t' - G_t \rrbracket$ for t yields $M_2 = M'_1 + \llbracket t' \rrbracket + (M_0 - M'_0) + \llbracket G - G_t \rrbracket$. By Condition 2c, $M'_1 \xrightarrow{a}$, so $M'_1 \xrightarrow{a} M'_2$ for some $M'_2 \in [M'_0]_{N'}$. As t' is the only transition in T' with $\ell'(t') = a$, we must have $M'_1[t']M'_2$. So $M'_1 + \llbracket t' \rrbracket = M'_2$. Since $\ell(G - G_t) \equiv \emptyset$ it follows that $M_2 \mathcal{B} M'_2$.
- (d) Follows directly from Condition 2b.
- (e) Follows directly from Condition 2a. □

The above is a variant of Lemma 15 that requires Condition b only for $U = U' = \emptyset$, and allows to conclude that N and N' are interleaving branching bisimilar (instead of branching ST-bisimilar) with explicit divergence. Likewise, the below is a variant of Theorem 9 that requires Condition 5 only for $U = U' = \emptyset$, and misses Condition 5g.

Theorem 10 Let $N = (S, T, F, M_0, \ell)$ and $N' = (S', T', F', M'_0, \ell')$ be finitary nets, with N' unlabelled, $S' \subseteq S$, and $M'_0 = M_0 \upharpoonright S'$. Suppose there exist sets $T_+ \subseteq T$ and $T_- \subseteq T$ and a class $NF \subseteq \mathbb{Z}^T$, such that

- 1–4. Conditions 1–4 from Theorem 9 hold, and

5. For every reachable marking $M' \in [M'_0]_{N'}$ there is an $H_{M'} \in \mathbb{N}^{T+}$ with $\ell(H_{M'}) \equiv \emptyset$, such that for each $H \in NF$ with $M := M' + (M_0 - M'_0) + \llbracket H \rrbracket \in [M_0]_N$ one has:

- (a) $M_{M'} := M' + (M_0 - M'_0) + \llbracket H_{M'} \rrbracket \in \mathbb{N}^S$,
- (b) if $M' \xrightarrow{a}$ with $a \in \text{Act}$ then $M_{M'} \xrightarrow{a}$,
- (c) $H \leq H_{M'}$,
- (d) if $H(u) < 0$ then $u \in T_-$,
- (e) if $H(u) < 0$ and $H(t) > 0$ then $\bullet u \cap \bullet t = \emptyset$,
- (f) if $H(u) < 0$ and $M[t]$ with $\ell(t) \neq \tau$ then $\bullet u \cap \bullet t = \emptyset$.

Then N and N' are interleaving branching bisimilar with explicit divergence.

Proof: A straightforward simplification of the proof of Theorem 9. \square

5.3 The Correctness Proof

We now apply the preceding theory to prove the correctness of the conflict replicating implementation.

Theorem 11 Let N' be a finitary unlabelled structural conflict net without a fully reachable pure \mathbf{M} . Then $\mathcal{I}(N') \approx_{bSTb}^{\Delta} N'$.

Proof: Let $N' = (S', T', F', M'_0, \ell')$ be the given finitary unlabelled structural conflict net without a fully reachable pure \mathbf{M} , and $N = (S, T, F, M_0, \ell)$ be its conflict replicated implementation $\mathcal{I}(N')$. This convention (at the expense of primes in the statement of the theorem) pays off in terms of a significant reduction in the number of primes in this thesis.

For future reference, Table 5.2 provides a place-oriented representation of the conflict replicating implementation of a given net $N' = (S', T', F', M'_0, \ell')$, with the macros for reversible transitions expanded. Here $T^{\leftarrow} = \{\text{initialise}_j \mid j \in T'\} \cup \{\text{transfer}_j^h \mid h <^\# j \in T'\}$, $(\text{transfer}_j^h)^{far} = \{\text{trans}_j^h\text{-out}\}$ and similarly $(\text{initialise}_j)^{far} = \{\text{pre}_k^j \mid k \geq^\# j\} \cup \{\text{trans}_j^h\text{-in} \mid h <^\# j\}$.

We will obtain Theorem 11 as an application of Theorem 9. Following the construction of N described in Section 5.1.3, we indeed have $S' \subseteq S$ and $M'_0 = M_0 \upharpoonright S'$. Let $T_+ \subseteq T$ be the set of transitions

$$\text{distribute}_p \quad \text{initialise}_j \cdot \text{fire} \quad \text{transfer}_j^h \cdot \text{fire} \quad (5.7)$$

Place	Pretransitions	arc weights	Posttransitions	arc weights	for all
p	$\{ \text{finalise}_i^t \}$	$F'(i, p)$	distribute_p (if $p^\bullet \neq \emptyset$)		$p \in S', i \in \bullet p$ $p \in S'$
p_c	$\{ \text{distribute}_p \}$		$\text{initialise}_c \cdot \text{fire}$	$F'(p, c)$	$p \in S', c \in p^\bullet$
π_c (marked)	$\{ \text{initialise}_c \cdot \text{reset}_i \}$	$F'(p, c)$	$\text{fetch}_{i,j}^{p,c}$	$F'(p, i)$	$j \geq^\# i \in p^\bullet$ $i \neq c \in T'$
pre_j^i	$\{ \text{initialise}_i \cdot \text{fire} \}$		execute_j^i		$j \geq^\# i \in T'$
$\text{trans}_j^{h\text{-in}}$	$\{ \text{execute}_j^i \}$		$\text{initialise}_i \cdot \text{undo}(\text{pre}_j^i)$		
$\text{trans}_j^{h\text{-out}}$	$\{ \text{initialise}_j \cdot \text{fire} \}$		$\text{transfer}_j^h \cdot \text{fire}$		$h <^\# j \in T'$
$\pi_{j\#l}$ (marked)	$\{ \text{transfer}_j^h \cdot \text{fire} \}$		$\text{initialise}_j \cdot \text{undo}(\text{trans}_j^{h\text{-in}})$		
$\text{fetch}_{i,j}^{p,c\text{-in}}$	$\{ \text{transfer}_j^h \cdot \text{fire} \}$		execute_j^i		$h <^\# j \in T', i \leq^\# j$
$\text{fetch}_{i,j}^{p,c\text{-out}}$	$\{ \text{execute}_j^i \}$		$\text{transfer}_j^i \cdot \text{undo}(\text{trans}_j^{h\text{-out}})$		
	$\{ \text{fetched}_j^i \}$		execute_j^i		$i \leq^\# j \leq^\# l \in T', c \neq l$
	$\{ \text{transfer}_l^j \cdot \text{reset}_c \text{ (if } j < l) \}$		$\text{transfer}_l^j \cdot \text{fire (if } j < l)$		
$\text{undo}_i(t)$	$\text{execute}_j^i \cdot \text{fire}$		$\text{fetch}_{i,j}^{p,c}$	$j \geq^\# i \in T', p \in \bullet i, c \in p^\bullet$	
$\text{reset}_i(t)$	fetched_j^i		$\text{fetch}_{i,j}^{p,c}$	$j \geq^\# i \in T', p \in \bullet i, c \in p^\bullet$	
$\text{ack}_i(t)$	$t \cdot \text{reset}_i, t \cdot \text{elide}_i$		fetched_j^i	$j \geq^\# i \in T', p \in \bullet i, c \in p^\bullet$	
$\text{fired}(t)$	$t \cdot \text{fire}$		$\text{undo}_i(t)$		$i \in T', t \in \Omega_i$
$\rho_i(t)$	$t \cdot \text{undo}_i$		$t \cdot \text{reset}_i$		$t \in T^{\leftarrow}, \Omega_i \ni t$
$\text{take}(f, t)$	$t \cdot \text{undo}_i$		$t \cdot \text{undo}(f)$		$t \in T^{\leftarrow}, \Omega_i \ni t, f \in t^{\text{far}}$
$\text{took}(f, t)$	$t \cdot \text{undo}(f)$		$t \cdot \text{undo}$		$t \in T^{\leftarrow}, f \in t^{\text{far}}$
$\rho(t)$	$t \cdot \text{undo}$		$t \cdot \text{reset}_i$		$t \in T^{\leftarrow}, \Omega_i \ni t$

Table 5.2: The conflict replicating implementation.

for any applicable values of $p \in S'$ and $h, j \in T'$. Furthermore, $T_- := (T \setminus (T_+ \cup \{\text{execute}_j^i \mid i \leq^\# j \in T'\}))$. We start with checking Conditions 1, 2 and 3 of Theorem 9.

1. Let $<_+$ be the partial order on T_+ given by the order of listing in (5.7) – so $\text{initialise}_i \cdot \text{fire} <_+ \text{transfer}_j^h \cdot \text{fire}$, for any $i \in T'$ and $h <^\# j \in T'$, but the transitions $\text{transfer}_j^h \cdot \text{fire}$ and $\text{transfer}_l^k \cdot \text{fire}$ for $(i, j) \neq (k, l)$ are unordered. By examining Table 5.2 we see that for any place with a pretransition t in T_+ , all its posttransitions u in T_+ appear higher in the $<_+$ -ordering: $t <_+ u$. From this it follows that $F \upharpoonright (S \cup T_+)$ is acyclic.
2. Let $<_-$ be the partial order on T_- given by the column-wise order of the following enumeration of T_- :

$t \cdot \text{undo}_i$	$\text{fetch}_{i,j}^{p,c}$
$\text{transfer}_j^h \cdot \text{undo}(f)$	fetched_j^i
$\text{transfer}_j^h \cdot \text{undo}$	$t \cdot \text{reset}_i$
$\text{initialise}_j \cdot \text{undo}(f)$	$t \cdot \text{elide}_i$
$\text{initialise}_j \cdot \text{undo}$	finalise_i^t

for any $t \in \{\text{initialise}_j, \text{transfer}_j^h\}$ and any applicable values of $f \in S, p \in S'$, and $h, i, j, c \in T'$. By examining Table 5.2 we see that for any place with a pretransition t in T_- , all its posttransitions u in T_- appear higher in the $<_-$ -ordering: $t <_- u$. From this it follows that $F \upharpoonright (S \cup T_-)$ is acyclic.

3. The only transitions $t \in T$ with $\ell(t) \neq \tau$ are execute_j^i , with $i \leq^\# j \in T'$. So take $i \leq^\# j \in T'$. Then the only transition $t' \in T'$ with $\ell'(t') = \ell(\text{execute}_j^i)$ is i . Now two statements regarding i and execute_j^i need to be proven. For the first, note that, for any $p \in \bullet i$, the places p, p_i and pre_j^i are faithful w.r.t. T_+ and $S' \cup \{s \in S \mid M_0(s) > 0\}$. Hence $p \text{ distribute}_p p_i \text{ initialise}_i \cdot \text{fire pre}_j^i \text{ execute}_j^i$ is a faithful path from p to execute_j^i . The arc weight of this path is $F'(p, i)$. Thus $\bullet i \leq^* \text{execute}_j^i$.

The second statement holds because, for all $i \leq^\# j \in T'$,

$$\begin{aligned} \llbracket i \rrbracket = & \llbracket \text{execute}_j^i \rrbracket + \sum_{p \in \bullet i} (F'(p, i) \cdot \text{distribute}_p + \sum_{c \in p^\bullet} \text{fetch}_{i,j}^{p,c}) + \\ & \text{fetched}_j^i + \text{finalise}_i + \sum_{t \in \Omega_i} t \cdot \text{elide}_i \rrbracket. \end{aligned} \quad (5.8)$$

To check that these equations hold, note that

$$\begin{aligned} \llbracket \text{distribute}_p \rrbracket &= -\{p\} + \{p_c \mid c \in p^\bullet\}, \\ \llbracket \text{execute}_j^i \rrbracket &= -\{\pi_{j\#l} \mid l \geq^\# j\} + \{\text{fetch}_{i,j}^{p,c}\text{-in} \mid p \in \bullet i, c \in p^\bullet\} \\ &\quad + \{\text{undo}_i(t) \mid t \in \Omega_i\}, \\ \llbracket \text{fetch}_{i,j}^{p,c} \rrbracket &= -\{\text{fetch}_{i,j}^{p,c}\text{-in}\} - F'(p, i) \cdot \{p_c\} + \{\text{fetch}_{i,j}^{p,c}\text{-out}\}, \\ \llbracket \text{fetched}_j^i \rrbracket &= -\{\text{fetch}_{i,j}^{p,c}\text{-out} \mid p \in \bullet i, c \in p^\bullet\} + \{\pi_{j\#l} \mid l \geq^\# j\} \\ &\quad + \{\text{reset}_i(t) \mid t \in \Omega_i\}, \\ \llbracket t \cdot \text{elide}_i \rrbracket &= -\{\text{undo}_i(t), \text{reset}_i(t) \mid t \in \Omega_i\} + \{\text{ack}_i(t) \mid t \in \Omega_i\}, \\ \llbracket \text{finalise}_i \rrbracket &= -\{\text{ack}_i(t) \mid t \in \Omega_i\} + \sum_{r \in i^\bullet} F'(i, r) \cdot \{r\}. \end{aligned}$$

Before we define the class $NF \subseteq \mathbb{Z}^T$ of signed multisets of transitions in normal form, and verify conditions 4 and 5, we derive some properties of the conflict replicating implementation $N = \mathcal{I}(N')$.

Claim 1 For any $M' \in \mathbb{Z}^{S'}$ and $G \in \mathbb{Z}^T$ with $M := M' + (M_0 - M'_0) + \llbracket G \rrbracket \in \mathbb{N}^S$ and for each $i \in T'$ and $t \in \Omega_i$ we have

$$G(t \cdot \text{elide}_i) + G(t \cdot \text{undo}_i) \leq \sum_{j \geq^\# i} G(\text{execute}_j^i) \quad (5.9)$$

$$G(\text{finalise}_i) \leq G(t \cdot \text{elide}_i) + G(t \cdot \text{reset}_i) \leq \sum_{j \geq^\# i} G(\text{fetched}_j^i) \quad (5.10)$$

$$G(t \cdot \text{reset}_i) \leq G(t \cdot \text{undo}_i). \quad (5.11)$$

Moreover, for each $t \in T^{\leftarrow}$ and $f \in t^{\text{far}}$,

$$\sum_{\{\omega \mid t \in \Omega_\omega\}} G(t \cdot \text{reset}_\omega) \leq G(t \cdot \text{undone}) \leq G(t \cdot \text{undo}(f)) \leq \sum_{\{\omega \mid t \in \Omega_\omega\}} G(t \cdot \text{undo}_\omega) \leq G(t \cdot \text{fire}) \quad (5.12)$$

and for each appropriate $c, h, i, j, l \in T'$ and $p \in S'$:

$$G(\text{fetched}_j^i) \leq G(\text{fetch}_{i,j}^{p,c}) \leq G(\text{execute}_j^i) \quad (5.13)$$

$$G(\text{initialise}_j \cdot \text{fire}) \leq 1 + \sum_{\omega} G(\text{initialise}_j \cdot \text{reset}_{\omega}) \quad (5.14)$$

$$G(\text{transfer}_j^h \cdot \text{fire}) - G(\text{transfer}_j^h \cdot \text{undone}) \leq G(\text{initialise}_j \cdot \text{fire}) - G(\text{initialise}_j \cdot \text{undo}(\text{trans}_j^h \cdot \text{in})) \quad (5.15)$$

$$G(\text{transfer}_l^j \cdot \text{fire}) + \sum_{i \leq \#j} G(\text{execute}_j^i) \leq 1 + \sum_{\omega} G(\text{transfer}_l^j \cdot \text{reset}_{\omega}) + \sum_{i \leq \#j} G(\text{fetched}_j^i) \quad (5.16)$$

$$\text{if } M[\text{execute}_j^i] \text{ then } 1 \leq G(\text{initialise}_i \cdot \text{fire}) - G(\text{initialise}_i \cdot \text{undo}(\text{pre}_j^i)) \quad (5.17)$$

$$\text{if } \exists i. M[\text{execute}_j^i] \text{ then } 1 \leq G(\text{transfer}_j^h \cdot \text{fire}) - G(\text{transfer}_j^h \cdot \text{undo}(\text{trans}_j^h \cdot \text{out})) \quad (5.18)$$

$$F'(p, c) \cdot (G(\text{initialise}_c \cdot \text{fire}) - G(\text{initialise}_c \cdot \text{undone})) + \sum_{j \geq \#i \in p} F'(p, i) \cdot G(\text{fetch}_{i,j}^{p,c}) \leq G(\text{distribute}_p) \quad (5.19)$$

$$G(\text{distribute}_p) \leq M'(p) + \sum_{\{i \in T' \mid p \in i \bullet\}} G(\text{finalise}_i). \quad (5.20)$$

Proof: For any $i \in T'$ and $t \in \Omega_i$, we have

$$M(\text{undo}_i(t)) = \left(\sum_{j \geq \#i} G(\text{execute}_j^i) \right) - G(t \cdot \text{elide}_i) - G(t \cdot \text{undo}_i) \geq 0,$$

given that $M'(\text{undo}_i(t)) = (M_0 - M'_0)(\text{undo}_i(t)) = \emptyset$. In this way, the place $\text{undo}_i(t)$ gives rise to the inequation (5.9) about G . Likewise, the places $\text{ack}_i(t)$, $\text{reset}_i(t)$ and $\rho_i(t)$, respectively, contribute (5.10) and (5.11), whereas $\rho(t)$, $\text{took}(t)$, $\text{take}(t)$ and $\text{fired}(t)$ yield (5.12). Note that for $j \neq l$, we have $G(\text{transfer}_j^l \cdot t) = 0$ regardless of t as no such transition exists in N . The remaining inequations then arise from $\text{fetch}_{i,j}^{p,c} \cdot \text{out}$, $\text{fetch}_{i,j}^{p,c} \cdot \text{in}$, π_j , $\text{trans}_j^h \cdot \text{in}$, $\pi_{j \neq l}$, pre_j^i , $\text{trans}_j^h \cdot \text{out}$, p_c and p , respectively. ■

(5.16) can be rewritten as $T_l^j + \sum_{i \leq \#j} E_j^i \leq 1$, where $T_l^j := G(\text{transfer}_l^j \cdot \text{fire}) - \sum_{\omega} G(\text{transfer}_l^j \cdot \text{reset}_{\omega})$ and $E_j^i := G(\text{execute}_j^i) - G(\text{fetched}_j^i)$. By (5.12) $\sum_{\omega} G(\text{transfer}_l^j \cdot \text{reset}_{\omega}) \leq G(\text{transfer}_l^j \cdot \text{fire})$, so $T_l^j \geq 0$, and likewise, by (5.13), $E_j^i \geq 0$ for all $i \leq \#j$. Hence, for all $i \leq \#j \leq \#l \in T'$,

$$0 \leq T_l^j \leq 1 \quad 0 \leq E_j^i \leq 1 \quad T_l^j + \sum_{i \leq \#j} E_j^i \leq 1. \quad (5.21)$$

In our next claim we study triples (M, M', G) with

$$(A) \quad M \in [M_0]_N, \quad M' \in [M'_0]_{N'} \text{ and } G \in \mathbb{Z}^T,$$

- (B) $M = M' + (M_0 - M'_0) + \llbracket G \rrbracket$,
- (C) $G(\text{finalise}^i) = 0$ for all $i \in T'$,
- (D) $G(\text{distribute}_p) \leq M'(p)$ for all $p \in S'$,
- (E) $G(\text{fetched}_l^k) \geq 0$ for all $k \leq^\# l \in T'$,
- (F) $G(\text{distribute}_p) \geq F'(p, i) \cdot G(\text{execute}_j^i)$ for all $i \leq^\# j \in T'$ and $p \in \bullet i$,
- (G) $0 \leq G(\text{execute}_j^i) \leq 1$ for all $i \leq^\# j \in T'$,
- (H) $G(\text{distribute}_p) \geq F'(p, j) \cdot G(\text{execute}_j^i)$ for all $i \leq^\# j \in T'$ and $p \in \bullet j$,
- (I) (in the notation of (5.21)) if $E_j^i = 1$ with $i \leq^\# j \in T'$ then $T_j^h = 1$ for all $h <^\# j$,
- (J) there are no $j \geq^\# i \not\leq^\# k \leq^\# l \in T'$ with $(i, j) \neq (k, \ell)$, $G(\text{execute}_j^i) > 0$ and $G(\text{execute}_l^k) > 0$,
- (K) there are no $i \leq^\# j \not\leq^\# k \leq^\# l \in T'$ with $(i, j) \neq (k, \ell)$, $G(\text{execute}_j^i) > 0$ and $G(\text{execute}_l^k) > 0$.

Given such a triple (M_1, M'_1, G_1) and a $t \in T$, we define $\text{next}(M_1, M'_1, G_1, t) =: (M, M', G)$ as follows: Let $G_2 := G_1 + \{t\}$. Take $M := M_1 + \llbracket t \rrbracket = M'_1 + (M_0 - M'_0) + \llbracket G_2 \rrbracket$. In case t is not of the form finalise^i we take $M' := M'_1 \in [M'_0]_{N'}$ and $G := G_2 \in \mathbb{Z}^T$. In case $t = \text{finalise}^i$ for some $i \in T'$ then $1 = G_2(\text{finalise}^i) \leq \sum_{j \geq^\# i} G_2(\text{execute}_j^i) = \sum_{j \geq^\# i} G_1(\text{execute}_j^i)$ from (C), (5.10) and (5.13), so by (G) and (J) there is a unique $j \geq^\# i$ with $G_1(\text{execute}_j^i) = 1$. We take $M' := M'_1 + \llbracket i \rrbracket$ and $G := G_2 - G_j^i$, where G_j^i is the right-hand side of (5.8).

Claim 2

- (1) If $M_1[t]$ and (M_1, M'_1, G_1) satisfies (A)-(K), then so does $\text{next}(M_1, M'_1, G_1, t)$.
- (2) For any $M \in [M_0]_N$ there exist M' and G such that (A)-(K) hold.

Proof: (2) follows from (1) via induction on the reachability of M . In case $M = M_0$ we take $M' := M'_0$ and $G := \emptyset$. Clearly, (A)-(K) are satisfied.

Hence we now show (1). Let $(M, M', G) := \text{next}(M_1, M'_1, G_1, t)$. We check that (M, M', G) satisfies the requirements (A)-(K).

- (A) By construction, $M \in [M_0]_N$ and $G \in \mathbb{Z}^T$. If t is not of the form finalise^i we have $M' = M_1 \in [M'_0]_{N'}$. Otherwise, by (D) and (F) we have $M'_1(p) \geq G_1(\text{distribute}_p) \geq F'(p, i)$ for all $p \in \bullet i$, and hence $M'_1[i] \geq 1$. This in turn implies that $M' = M'_1 + \llbracket i \rrbracket \in [M'_0]_{N'}$.
- (B) In case t is not of the form finalise^i we have

$$M = M_1 + \llbracket t \rrbracket = M'_1 + (M_0 - M'_0) + \llbracket G_1 + t \rrbracket = M' + (M_0 - M'_0) + \llbracket G \rrbracket.$$

In case $t = \text{finalise}^i$ we have $M = M'_1 + (M_0 - M'_0) + \llbracket G_2 \rrbracket = M' + (M_0 - M'_0) + \llbracket G \rrbracket$, using that $\llbracket i \rrbracket = \llbracket G_j^i \rrbracket$.

- (C) In case $t = \text{finalise}^i$ we have $G(\text{finalise}^i) = G_1(\text{finalise}^i) + 1 - G_j^i(\text{finalise}^i) = 0 + 1 - 1 = 0$.
Otherwise $G(\text{finalise}^i) = G_1(\text{finalise}^i) + 0 = 0 + 0 = 0$.
- (D) This follows immediately from (C) and (5.20).
- (E) The only time that this invariant is in danger is when $t = \text{finalise}^i$. Then $G = G_1 + \{\text{finalise}^i\} - G_j^i$ for a certain $j \geq^\# i$ with $G_1(\text{execute}_j^i) = 1$. By (J)⁴ $G_1(\text{execute}_l^i) \leq 0$ for all $l \geq^\# i$ with $l \neq j$. Hence by (5.13) $G_1(\text{fetched}_l^i) \leq 0$ for all such l . By (C) $G_2(\text{finalise}^i) = G_1(\text{finalise}^i) + 1 = 1$, so by (5.10) $\sum_{l \geq^\# i} G_1(\text{fetched}_l^i) = \sum_{l \geq^\# i} G_2(\text{fetched}_l^i) > 0$; hence it must be that $G_1(\text{fetched}_j^i) > 0$. By (E)⁴ $G_1(\text{fetched}_l^k) \geq 0$ for all $k \leq^\# l \in T'$. Given that $G_j^i(\text{fetched}_j^i) = 1$ and $G_j^i(\text{fetched}_l^k) = 0$ for all $(k, l) \neq (i, j)$, we obtain $G(\text{fetched}_l^k) \geq 0$ for all $k \leq^\# l \in T'$.
- (F) Take $i \leq^\# j \in T'$ and $p \in \bullet i$. There are two occasions where the invariant is in danger: when $t = \text{execute}_j^i$ and when $t = \text{finalise}^k$ with $k \in T'$. First let $t = \text{execute}_j^i$. Then $M_1[\text{execute}_j^i]$. Thus,

$$\begin{aligned}
& G(\text{distribute}_p) \\
& \geq F'(p, i) \cdot (G(\text{initialise}_i \cdot \text{fire}) - G(\text{initialise}_i \cdot \text{undone})) \\
& \quad + \sum_{h \geq^\# g \in p \bullet} F'(p, g) \cdot G(\text{fetch}_{g,h}^{p,i}) \\
& \geq F'(p, i) \cdot (G(\text{initialise}_i \cdot \text{fire}) - G(\text{initialise}_i \cdot \text{undone})) \\
& \quad + \sum_{h \geq^\# g \in p \bullet} F'(p, g) \cdot G(\text{fetched}_h^g) \\
& \geq F'(p, i) \cdot (G(\text{initialise}_i \cdot \text{fire}) - G(\text{initialise}_i \cdot \text{undone})) \\
& \quad + F'(p, i) \cdot G(\text{fetched}_j^i) \\
& \geq F'(p, i) \cdot ((G(\text{initialise}_i \cdot \text{fire}) - G(\text{initialise}_i \cdot \text{undo}(\text{pre}_j^i))) \\
& \quad + G(\text{fetched}_j^i)) \\
& \geq F'(p, i) \cdot (1 + G(\text{fetched}_j^i)) \\
& \geq F'(p, i) \cdot G(\text{execute}_j^i)
\end{aligned}$$

by (5.19), (5.13), (E), (5.12), (5.17) and (5.21), respectively. By (5.12) $G(\text{initialise}_i \cdot \text{fire}) - G(\text{initialise}_i \cdot \text{undone}) \geq 0$. So by (5.19), (E), and (5.13) $G(\text{distribute}_p) \geq 0$. For this reason we may assume, w.l.o.g., that $G(\text{execute}_j^i) \geq 1$.

In the other case have $G = G_1 + \{\text{finalise}^k\} - G_l^k$ for certain $l \geq^\# k$ with $G_1(\text{execute}_l^k) = 1$. Since $G_j^i(\text{execute}_j^i) \geq 0$, we also have $G_1(\text{execute}_j^i) \geq 1$ as explained under the construction of *next*. Using (J) this implies that $\neg(i \stackrel{\#}{=} k)$ or $(i, j) = (k, l)$. In the latter case $G(\text{execute}_j^i) = G_1(\text{execute}_j^i) - G_j^i(\text{execute}_j^i) = 1 - 1 = 0$, contradicting our assumption. In the former case $p \notin \bullet k$, so $G_l^k(\text{distribute}_p) = 0$ and hence $G(\text{distribute}_p) = G_1(\text{distribute}_p) \geq F'(p, i) \cdot G_1(\text{execute}_j^i) = F'(p, i) \cdot G(\text{execute}_j^i)$.

⁴We use (J) and (E) for G_1 only, making use of the induction hypothesis.

- (G) That $G(\text{execute}_j^i) \geq 0$ follows from (E) and (5.13). If $G(\text{execute}_j^i) \geq 2$ for some $i \leq^\# j \in T'$ then $M'(p) \geq G(\text{distribute}_p) \geq 2 \cdot F'(p, i)$ for all $p \in \bullet i$, using (D) and (F), so $M'[2 \cdot \{i\}]_{N'}$. Since N' is a finitary structural conflict net, it has no self-concurrency, so this is impossible.
- (H) Take $i \leq^\# j \in T'$ and $p \in \bullet j$. The case $i = j$ follows from (F), so assume $i <^\# j$. By (5.12) we have $G(\text{initialise}_i \cdot \text{fire}) - G(\text{initialise}_i \cdot \text{undone}) \geq 0$. So by (5.19), (E), and (5.13) $G(\text{distribute}_p) \geq 0$. Hence, using (G), we may assume, w.l.o.g., that $G(\text{execute}_j^i) = 1$. We need to investigate the same two cases as in the proof of (F) above. First let $t = \text{execute}_j^i$. Then $M_1[\text{execute}_j^i]$. Thus,

$$\begin{aligned}
& G(\text{distribute}_p) \\
& \geq F'(p, j) \cdot (G(\text{initialise}_j \cdot \text{fire}) - G(\text{initialise}_j \cdot \text{undone})) \quad (\text{by (5.19)}) \\
& \quad + \sum_{h \geq^\# g \in p \bullet} F'(p, g) \cdot G(\text{fetch}_{g,h}^{p,j}) \\
& \geq F'(p, j) \cdot \left(\begin{array}{c} G(\text{initialise}_j \cdot \text{fire}) \\ -G(\text{initialise}_j \cdot \text{undone}) \end{array} \right) \quad (\text{by (E) and (5.13)}) \\
& \geq F'(p, j) \cdot \left(\begin{array}{c} G(\text{initialise}_j \cdot \text{fire}) \\ -G(\text{initialise}_j \cdot \text{undo}(\text{trans}_j^i\text{-in})) \end{array} \right) \quad (\text{by (5.12)}) \\
& \geq F'(p, j) \cdot (G(\text{transfer}_j^i \cdot \text{fire}) - G(\text{transfer}_j^i \cdot \text{undone})) \quad (\text{by (5.15)}) \\
& \geq F'(p, j) \cdot \left(\begin{array}{c} G(\text{transfer}_j^i \cdot \text{fire}) \\ -G(\text{transfer}_j^i \cdot \text{undo}(\text{trans}_j^i\text{-out})) \end{array} \right) \quad (\text{by (5.12)}) \\
& \geq F'(p, j) \quad (\text{by (5.18)}).
\end{aligned}$$

Now let $t = \text{finalise}^k$ with $k \in T'$. We have $G = G_1 + \{\text{finalise}^k\} - G_l^k$ for certain $l \geq^\# k$ with $G_1(\text{execute}_l^k) = 1$. Since $G_j^i(\text{execute}_j^i) \geq 0$, we also have $G_1(\text{execute}_j^i) \geq 1$. By (K) this implies that $\neg(j \stackrel{\#}{=} k)$ or $(i, j) = (k, l)$. In the latter case $G(\text{execute}_j^i) = G_1(\text{execute}_j^i) - G_j^i(\text{execute}_j^i) = 1 - 1 = 0$, contradicting our assumption. In the former case $p \notin \bullet k$, so $G_l^k(\text{distribute}_p) = 0$ and hence $G(\text{distribute}_p) = G_1(\text{distribute}_p) \geq F'(p, j) \cdot G_1(\text{execute}_j^i) = F'(p, j) \cdot G(\text{execute}_j^i)$.

- (I) Let $i \leq^\# j \in T'$ and $h <^\# j$. Since, for all $k \leq^\# l \in T'$, $G_l^k(\text{transfer}_j^h \cdot \text{fire}) = \sum_\omega G_l^k(\text{transfer}_j^h \cdot \text{reset}_\omega) = 0$ and $G_l^k(\text{execute}_j^i) = G_l^k(\text{fetched}_j^i)$, the invariant is preserved when t has the form finalise^b . Using (5.21), it is in danger only when $t = \text{execute}_j^i$ or $t = \text{transfer}_j^h \cdot \text{reset}_\omega$ for some ω with $\text{transfer}_j^h \in \Omega_\omega$.

First assume $M_1[\text{execute}_j^i]$ and $T_j^h = G_1(\text{transfer}_j^h \cdot \text{fire}) - \sum_\omega G_1(\text{transfer}_j^h \cdot \text{reset}_\omega) = 0$. Then

$$\begin{aligned}
1 & \leq G_1(\text{transfer}_j^h \cdot \text{fire}) - G_1(\text{transfer}_j^h \cdot \text{undo}(\text{trans}_j^h\text{-out})) \quad (\text{by (5.18)}) \\
& \leq G_1(\text{transfer}_j^h \cdot \text{fire}) - \sum_\omega G_1(\text{transfer}_j^h \cdot \text{reset}_\omega) = 0 \quad (\text{by (5.12)}),
\end{aligned}$$

which is a contradiction.

Next assume $t = \text{transfer}_j^h \cdot \text{reset}_k$ with $k \neq j$, and $E_j^i = 1$. By (E) and (G) the latter implies that $G_1(\text{execute}_j^i) = 1$ and $G_1(\text{fetched}_j^i) = 0$. Then

$$\begin{aligned} 0 &= G_1(\text{finalise}_k^i) && \text{(by (C))} \\ &\leq G_1(\text{transfer}_j^h \cdot \text{elide}_k) + G_1(\text{transfer}_j^h \cdot \text{reset}_k) && \text{(by (5.10))} \\ &< G(\text{transfer}_j^h \cdot \text{elide}_k) + G(\text{transfer}_j^h \cdot \text{reset}_k) \\ &\leq \sum_{l \geq \#k} G(\text{fetched}_l^k) && \text{(by (5.10)).} \end{aligned}$$

Hence $G_1(\text{fetched}_l^k) = G(\text{fetched}_l^k) > 0$ for some $l \geq \#k$, and by (5.13) also $G_1(\text{execute}_l^k) > 0$. Using (K) we obtain $(i, j) = (k, l)$, thereby obtaining a contradiction ($0 = G_1(\text{fetched}_j^i) = G_1(\text{fetched}_l^k) > 0$).

- (J) Let $j \geq \#i \neq k \leq \#l \in T'$ with $(i, j) \neq (k, l)$. The invariant is in danger only when $t = \text{execute}_j^i$ or $t = \text{execute}_l^k$. W.l.o.g. let $t = \text{execute}_l^k$, with $G_1(\text{execute}_l^k) = 0$ and $G_1(\text{execute}_j^i) \geq 1$.

Making a case distinction, first assume $G(\text{fetched}_j^i) \geq 1$. Using (D), (F) and that $G(\text{execute}_l^k) = 1$, $M'(p) \geq G(\text{distribute}_p) \geq F'(p, k)$ for all $p \in \bullet k$. Likewise, $M'(p) \geq G(\text{distribute}_p) \geq F'(p, i)$ for all $p \in \bullet i$. Moreover, just as in the proof of (F), we derive, for all $p \in \bullet i \cap \bullet k$,

$$\begin{aligned} M'(p) &\geq G(\text{distribute}_p) \\ &\geq F'(p, k) \cdot (G(\text{initialise}_k \cdot \text{fire}) - G(\text{initialise}_k \cdot \text{undone})) \\ &\quad + \sum_{h \geq \#g \in p \bullet} F'(p, g) \cdot G(\text{fetch}_{g,h}^{p,k}) \\ &\geq F'(p, k) \cdot (G(\text{initialise}_k \cdot \text{fire}) - G(\text{initialise}_k \cdot \text{undone})) \\ &\quad + \sum_{h \geq \#g \in p \bullet} F'(p, g) \cdot G(\text{fetched}_h^g) \\ &\geq F'(p, k) \cdot (G(\text{initialise}_k \cdot \text{fire}) - G(\text{initialise}_k \cdot \text{undone})) \\ &\quad + F'(p, i) \cdot G(\text{fetched}_j^i) \\ &\geq F'(p, k) \cdot (G(\text{initialise}_k \cdot \text{fire}) - G(\text{initialise}_k \cdot \text{undo}(\text{pre}_l^k))) \\ &\quad + F'(p, i) \cdot G(\text{fetched}_j^i) \\ &\geq F'(p, k) + F'(p, i) \end{aligned}$$

by (D), (5.19), (5.13), (E), (5.12) and (5.17), respectively. It follows that $M'[\{k\} + \{i\}]$. As $i \neq k$ and N' is a finitary structural conflict net, this is impossible. (Note that this argument holds regardless whether $i = k$.)

Now assume $G(\text{fetched}_j^i) \leq 0$. Then, in the notation of (5.21), $E_j^i = 1$. As $G_1(\text{execute}_l^k) = 0$, (E) and (5.13) yield $G_1(\text{fetched}_l^k) = 0$. Hence we have $G(\text{execute}_l^k) = 1$ and $G(\text{fetched}_l^k) = 0$, so $E_l^k = 1$. We will conclude the proof by deriving a contradiction from $E_j^i = E_l^k = 1$. In case $j = l$ this contradiction emerges immediately from (5.21). By symmetry it hence suffices to consider the case $j < l$.

By (D) and (H) we have $M'(p) \geq G(\text{distribute}_p) \geq F'(p, j)$ for all $p \in \bullet j$, so $M'[j]$. Likewise $M'[l]$ and, using (F), $M'[i]$ and $M'[k]$. Since $j \neq i \neq k$ and N' has no fully reachable visible pure M, $j \neq k$. Since $j \neq k \neq l$ and N' has no fully reachable visible pure M, $j \neq l$. So $j < \#l$. By (5.21), using that $E_j^i = 1$, $T_l^j = 0$. This is in contradiction with $E_l^k = 1$ and (I).

- (K) Suppose that $G(\text{execute}_j^i) > 0$ and $G(\text{execute}_l^k) > 0$, with $i \leq^\# j \stackrel{\#}{=} k \leq^\# l \in T'$. By (D) and (H) we have $M'(p) \geq G(\text{distribute}_p) \geq F'(p, j)$ for all $p \in \bullet j$, so $M'[j]$. Likewise, using (F), $M'[i]$ and $M'[k]$. Since $i \stackrel{\#}{=} j \stackrel{\#}{=} k$ and N' has no fully reachable visible pure M, $i \stackrel{\#}{=} k$. Using this, the result follows from (J). ■

Claim 3 For any $M \in [M_0]_N$ there exist $M' \in [M'_0]_{N'}$ and $G \in \mathbb{Z}^T$ satisfying (A)–(K) from Claim 2, and

- (L) there are no $j \geq^\# i \stackrel{\#}{=} k \leq^\# l \in T'$ with $M[\text{execute}_j^i]$ and $G(\text{execute}_l^k) > 0$,
 (M) there are no $i \leq^\# j \stackrel{\#}{=} k \leq^\# l \in T'$ with $M[\text{execute}_j^i]$ and $G(\text{execute}_l^k) > 0$,
 (N) if $M[\text{execute}_j^i]$ for $i \leq^\# j \in T'$ then $M'[j]$.

Proof: Given M , by Claim 2(2) there are M' and G so that the triple (M, M', G) satisfies (A)–(K). Assume $M[\text{execute}_j^i]$ for some $i \leq^\# j \in T'$. Let $M_1 := M + \llbracket \text{execute}_j^i \rrbracket$ and $G_1 := G + \{\text{execute}_j^i\}$. By (G) $G(\text{execute}_j^i) \geq 0$, and hence $G_1(\text{execute}_j^i) > 0$. By Claim 2(1) the triple (M_1, M', G_1) satisfies (A)–(K).

- (L) Suppose $G(\text{execute}_l^k) > 0$ for certain $l \geq^\# k \stackrel{\#}{=} i$. In case $(i, j) = (k, l)$, $G_1(\text{execute}_j^i) \geq 2$, contradicting (G). In case $(i, j) \neq (k, l)$, G_1 fails (J), also a contradiction.
 (M) Suppose $G(\text{execute}_l^k) > 0$ for certain $l \geq^\# k \stackrel{\#}{=} j$. Then G_1 fails (G) or (K), a contradiction.
 (N) By (D) and (H) $M'(p) \geq G_1(\text{distribute}_p) \geq F'(p, j)$ for all $p \in \bullet j$, so $M'[j]$. ■

Claim 4 If $M[\{\text{execute}_j^i\} + \{\text{execute}_l^k\}]$ for some $M \in [M_0]_N$ then $\neg(i \stackrel{\#}{=} k)$.

Proof: Suppose $M[\{\text{execute}_j^i\} + \{\text{execute}_l^k\}]$ for some $M \in [M_0]_N$. Then by Claim 2(2) there exist $M' \in [M'_0]_{N'}$ and $G \in \mathbb{Z}^T$ satisfying (A)–(K). Let $M_1 := M + \llbracket \text{execute}_l^k \rrbracket$ and $G_1 := G + \{\text{execute}_l^k\}$. By Claim 2(1) the triple (M_1, M', G_1) satisfies (A)–(K). Let $M_2 := M_1 + \llbracket \text{execute}_j^i \rrbracket$ and let $G_2 := G_1 + \{\text{execute}_j^i\}$. Again by Claim 2(1), also the triple (M_2, M', G_2) satisfies (A)–(K). By (G) $G(\text{execute}_j^i) \geq 0$, so in case $(i, j) = (k, l)$ we obtain $G_2(\text{execute}_j^i) \geq 2$, contradicting (G). Hence $(i, j) \neq (k, l)$. Moreover, we have $G_2(\text{execute}_l^k) > 0$ and $G_2(\text{execute}_j^i) > 0$. Now (J) implies $\neg(i \stackrel{\#}{=} k)$. ■

For any $t \in \{\text{initialise}_j, \text{transfer}_j^h\}$ with $h, j \in T'$, and any $\omega \in \Omega$ with $t \in \Omega_\omega$, we write

$$t(\omega) := t \cdot \text{fire} + t \cdot \text{undo}_\omega + \left(\sum_{f \in t^{\text{far}}} t \cdot \text{undo}(f) \right) + t \cdot \text{undone} + t \cdot \text{reset}_\omega.$$

The transition t has no preplaces of type *in*, nor postplaces of type *out*. By checking in Table 5.1 or Figure 5.3 that each other place occurs as often in

$\bullet u(\omega) + (u \cdot \text{elide}_\omega)^\bullet$ as in $u(\omega)^\bullet + \bullet(u \cdot \text{elide}_\omega)$, one verifies, for any $\omega \in \Omega$ with $t \in \Omega_\omega$, that

$$\llbracket t(\omega) \rrbracket = \llbracket t \cdot \text{elide}_\omega \rrbracket. \quad (5.22)$$

Let \equiv be the congruence relation on finite signed multisets of transitions generated by

$$t(\omega) \equiv t \cdot \text{elide}_\omega \quad (5.23)$$

for all $t \in \{\text{initialise}_j, \text{transfer}_j^h \mid h, j \in T'\}$ and $\omega \in \Omega$ with $\Omega_\omega \ni t$. Here *congruence* means that $G_1 \equiv G_2$ implies $k \cdot G_1 \equiv k \cdot G_2$ and $G_1 + H \equiv G_2 + H$ for all $k \in \mathbb{Z}$ and $H \in \mathbb{Z}^T$. Using (5.22) $G_1 \equiv G_2$ implies $\llbracket G_1 \rrbracket = \llbracket G_2 \rrbracket$.

Claim 5 If $M' = \llbracket G \rrbracket$ for $M' \in \mathbb{Z}^{S'}$ and $G \in \mathbb{Z}^T$ such that for all $i \in T'$ we have $G(\text{finalise}^i) = 0$ and either $\forall j \geq^\# i. G(\text{execute}_j^i) \geq 0$ or $\forall j \geq^\# i. G(\text{execute}_j^i) \leq 0$, then $G \equiv \emptyset$.

Proof: Let M' and G be as above. W.l.o.g. we assume $G(t \cdot \text{elide}_\omega) = 0$ for all $t \in \{\text{initialise}_j, \text{transfer}_j^h\}$ and all $\omega \in \Omega$ with $t \in \Omega_\omega$, for any G can be brought into that form by applying (5.23). For each $s \in S \setminus S'$ we have $M'(s) = 0$, and using this the inequations (5.9)–(5.13) and (5.19) of Claim 1 turn into equations. For each $i \in T'$ we have $G(\sum_{j \geq^\# i} \text{execute}_j^i) = 0$, using (the equational form of) (5.9)–(5.11), and that $G(\text{finalise}^i) = 0$. Since $G(\text{execute}_j^i) \geq 0$ (or ≤ 0) for all $j \geq^\# i$, this implies that $G(\text{execute}_j^i) = 0$ for each $i \leq^\# j \in T'$. With (5.13) we obtain $G(\text{fetched}_j^i) = G(\text{fetch}_{i,j}^{p,c}) = 0$ for each applicable p, c, i, j . Using that $G(t \cdot \text{elide}_\omega) = 0$ for each applicable t and ω , with (5.10)–(5.12) and (5.19) we find $G(t) = 0$ for all $t \in T$. ■

Claim 6 Let $M := M' + (M_0 - M'_0) + \llbracket H \rrbracket \in [M_0]_N$ for $M' \in [M'_0]_{N'}$ and $H \in \mathbb{Z}^T$ with $H(\text{execute}_j^i) = 0$ for all $i \leq^\# j \in T'$.

- (a) If $H(\text{finalise}^i) < 0$ and $H(\text{finalise}^k) < 0$ for certain $i, k \in T'$ then $\neg(i \# k)$.
- (b) If $M[\text{execute}_j^i] > 0$ and $H(\text{finalise}^k) < 0$ for certain $i, k \in T'$ then $\neg(i \stackrel{\#}{=} k)$ and $\neg(j \stackrel{\#}{=} k)$.
- (c) $H(\text{distribute}_p) \geq 0$ for all $p \in S'$ (with $p^\bullet \neq \emptyset$).
- (d) Let $c \stackrel{\#}{=} i \in T'$. If $\forall p \in \bullet c. H(\text{distribute}_p) \geq F'(p, c)$, then $H(\text{finalise}^i) = 0$.
- (e) If $M[\text{execute}_j^i] > 0$ with $i \leq^\# j \in T'$ then $M'[j]$.

Proof: By Claim 3 there exist $M'_1 \in [M'_0]_{N'}$ and $G_1 \in \mathbb{Z}^T$ satisfying (B)–(N) (with M, M'_1 and G_1 playing the rôles of M, M' and G). In particular, $M = M'_1 + (M_0 - M'_0) + \llbracket G_1 \rrbracket$, $G_1(\text{finalise}^i) = 0$ for all $i \in T'$, and $G_1(\text{execute}_j^i) \geq 0$ for all $i \leq^\# j \in T'$. Using (J), for each $i \in T'$ there is at most one $j \geq^\# i$ with $G_1(\text{execute}_j^i) > 0$; we denote this j by $f(i)$, and let $f(i) := i$ when there is no such j . This makes $f : T' \rightarrow T'$ a function, satisfying $G_1(\text{execute}_j^i) = 0$ for all $j \geq^\# i$ with $j \neq f(i)$.

Given that $H(\text{execute}_j^i) = 0$ for all $i \leq^\# j \in T'$, (5.9)–(5.11) (or (5.10) and (5.13)) imply $H(\text{finalise}^i) \leq 0$ for all $i \in T'$. Let $M'_2 := M' + \sum_{i \in T'} (H(\text{finalise}^i) \cdot \llbracket i \rrbracket)$ and $G_2 := H - \sum_{i \in T'} H(\text{finalise}^i) \cdot G_{f(i)}^i$, where G_j^i is the right-hand side of (5.8). Then $M = M' + (M_0 - M'_0) + \llbracket H \rrbracket = M'_2 + (M_0 - M'_0) + \llbracket G_2 \rrbracket$, using that $\llbracket i \rrbracket = \llbracket G_{f(i)}^i \rrbracket$. Moreover, $G_2(\text{finalise}^i) = 0$ for all $i \in T'$, using that $G_{f(i)}^i(\text{finalise}^i) = 1$.

It follows that $M'_1 - M'_2 = \llbracket G_2 - G_1 \rrbracket$. Moreover, we have $(G_2 - G_1)(\text{finalise}^i) = 0$ for all $i \in T'$. We proceed to show that $G_2 - G_1$ satisfies the remaining precondition of Claim 5. So let $i \in T'$. In case $H(\text{finalise}^i) = 0$, for all $j \geq^\# i$ we have $G_2(\text{execute}_j^i) = 0$, and $G_1(\text{execute}_j^i) \geq 0$ by (G). Hence $(G_2 - G_1)(\text{execute}_j^i) \leq 0$. In case $H(\text{finalise}^i) < 0$, we have $G_2(\text{execute}_{f(i)}^i) \geq 1$, and hence, using (G), $(G_2 - G_1)(\text{execute}_{f(i)}^i) \geq 0$. Furthermore, for all $j \neq f(i)$, $G_2(\text{execute}_j^i) \geq 0$ and $G_1(\text{execute}_j^i) = 0$, so again $(G_2 - G_1)(\text{execute}_j^i) \geq 0$.

Thus we may apply Claim 5, which yields $G_2 \equiv G_1$. It follows that $M'_2 = M'_1 \in [M'_0]_{N'}$.

- (a) Suppose that $H(\text{finalise}^i) < 0$ and $H(\text{finalise}^k) < 0$ for certain $i \neq k \in T'$. Then $G_2(\text{execute}_{f(i)}^i) > 0$ and $G_2(\text{execute}_{f(k)}^k) > 0$, so $G_1(\text{execute}_{f(i)}^i) > 0$ and $G_1(\text{execute}_{f(k)}^k) > 0$, contradicting (J).
- (b) Suppose that $M[\text{execute}_j^i]$ and $H(\text{finalise}^k) < 0$ for certain $k \neq i$ or $k \neq j$. Then $G_1(\text{execute}_{f(k)}^k) = G_2(\text{execute}_{f(k)}^k) > 0$, contradicting (L) or (M).
- (c) By (a), for any given $p \in S'$ there is at most one $i \in p^\bullet$ with $H(\text{finalise}^i) < 0$. For all $i \in T'$ with $i \notin p^\bullet$ we have $G_{f(i)}^i(\text{distribute}_p) = 0$. First suppose $k \in p^\bullet$ satisfies $H(\text{finalise}^k) < 0$. Then

$$\begin{aligned} G_1(\text{execute}_{f(k)}^k) &= G_2(\text{execute}_{f(k)}^k) \\ &= H(\text{execute}_{f(k)}^k) - \sum_{i \in T'} H(\text{finalise}^i) \cdot G_{f(i)}^i(\text{execute}_{f(k)}^k) \\ &= 0 - H(\text{finalise}^k), \end{aligned}$$

so by (F) $G_1(\text{distribute}_p) \geq -F'(p, k) \cdot H(\text{finalise}^k)$. Hence

$$\begin{aligned} H(\text{distribute}_p) &= G_2(\text{distribute}_p) + \sum_{i \in T'} H(\text{finalise}^i) \cdot G_{f(i)}^i(\text{distribute}_p) \\ &= G_1(\text{distribute}_p) + H(\text{finalise}^k) \cdot G_{f(k)}^k(\text{distribute}_p) \\ &\geq -F'(p, k) \cdot H(\text{finalise}^k) + H(\text{finalise}^k) \cdot F'(p, k) = 0. \end{aligned}$$

In case there is no $i \in p^\bullet$ with $H(\text{finalise}^i) < 0$ we have

$$\begin{aligned} H(\text{distribute}_p) &= G_2(\text{distribute}_p) + \sum_{i \in T'} H(\text{finalise}^i) \cdot G_{f(i)}^i(\text{distribute}_p) \\ &= G_1(\text{distribute}_p) \geq 0 \end{aligned}$$

by (F) and (G).

- (d) Since $H(\text{finalise}^i) \leq 0$ and $G_{f(i)}^i(\text{distribute}_p) \geq 0$ for all $i \in T'$, also using (c), all summands in $H(\text{distribute}_p) + \sum_{i \in T'} -H(\text{finalise}^i) \cdot G_{f(i)}^i(\text{distribute}_p)$ are

positive. Now suppose $H(\text{finalise}^i) < 0$ for certain $i \in T'$. Then, using (D), for all $p \in \bullet i$,

$$M'_1(p) \geq G_1(\text{distribute}_p) = G_2(\text{distribute}_p) \geq G_{f(i)}^i(\text{distribute}_p) = F'(p, i).$$

Furthermore, let $c \# i$ and suppose $H(\text{distribute}_p) \geq F'(p, c)$ for all $p \in \bullet c$. Then, using (D),

$$M'_1(p) \geq G_1(\text{distribute}_p) = G_2(\text{distribute}_p) \geq H(\text{distribute}_p) \geq F'(p, c)$$

for all $p \in \bullet c$. Moreover, if $p \in \bullet c \cap \bullet i$ then

$$\begin{aligned} M'_1(p) &\geq G_2(\text{distribute}_p) \\ &\geq H(\text{distribute}_p) + G_{f(i)}^i(\text{distribute}_p) \\ &\geq F'(p, c) + F'(p, i). \end{aligned}$$

Hence $M'_2[\{c\} + \{i\}]$. However, since $c \# i$ and N' is a structural conflict net, this is impossible.

- (e) Suppose $M[\text{execute}_j^i]$ with $i \leq \# j \in T'$. Then $M'_1[j]$ by (N).
Now $M' = M'_1 + \sum_{k \in T'} -H(\text{finalise}^k) \cdot \llbracket k \rrbracket$, with $-H(\text{finalise}^k) \geq 0$ for all $k \in T'$. Whenever $-H(\text{finalise}^k) > 0$ then $\neg(j \# k)$ by (b). Hence $M'[j]$. ■

We now define the class $NF \subseteq \mathbb{Z}^T$ of signed multisets of transitions in *normal form* by $H \in NF$ iff $\ell(H) \equiv \emptyset$ and, for all $t \in \{\text{initialise}_j, \text{transfer}_j^h \mid h, j \in T'\}$:

- (NF-1) $H(t \cdot \text{elide}_\omega) \leq 0$ for each $\omega \in \Omega$,
 (NF-2) $H(t \cdot \text{undo}_\omega) \geq 0$ for each $\omega \in \Omega$, or $H(t \cdot \text{fire}) \geq 0$,
 (NF-3) and if $H(t \cdot \text{elide}_\omega) < 0$ for any $\omega \in \Omega$, then $H(t \cdot \text{undo}_\omega) \leq 0$ and $H(t \cdot \text{fire}) \leq 0$.

We proceed verifying the remaining conditions of Theorem 9.

4. By applying (5.23), each signed multiset $G \in \mathbb{Z}^T$ with $\ell(G) \equiv \emptyset$ can be converted into a signed multiset $H \in NF$ with $\ell(H) \equiv \emptyset$, such that $\llbracket H \rrbracket = \llbracket G \rrbracket$. Namely, for any $t \in \{\text{initialise}_j, \text{transfer}_j^h \mid h, j \in T'\}$, first of all perform the following three transformations, until none is applicable:

- (i) correct a positive count of a transition $t \cdot \text{elide}_\omega$ in G by adding $t(\omega) - t \cdot \text{elide}_\omega$ to G ;
- (ii) if both $H(t \cdot \text{undo}_\omega) < 0$ for some ω and $H(t \cdot \text{fire}) < 0$, correct this in the same way;
- (iii) and if, for some ω , $t \cdot \text{elide}_\omega$ has a negative and $t \cdot \text{undo}_\omega$ a positive count, add $t \cdot \text{elide}_\omega - t(\omega)$.

Note that transformation (iii) will never be applied to the same ω as (i) or (ii), so termination is ensured. Properties (NF-1) and (NF-2) then hold for t . After termination of (i)–(iii), perform

(iv) if, for some ω , $H(t \cdot \text{elide}_\omega) < 0$ and $H(t \cdot \text{fire}) > 0$, add $t \cdot \text{elide}_\omega - t(\omega)$.

This will ensure that also (NF-3) is satisfied, while preserving (NF-1) and (NF-2).

Define the function $f : T \rightarrow \mathbb{N}$ by $f(u) := 1$ for all $u \in T$ not of the form $u = t \cdot \text{elide}_\omega$, and $f(t \cdot \text{elide}_\omega) := f(t(\omega))$ (extending f to multisets as in Definition 1). Then surely $f(G) = f(H)$.

5. Let $M' \in \mathbb{N}^{S'}$, $U' \in \mathbb{N}^{T'}$ and $U \in \mathbb{N}^T$ with $\ell(U) = \ell'(U')$ and $M' + \bullet U' \in [M'_0]_{N'}$. Since N' is a finitary structural conflict net, it admits no self-concurrency, so, as $\bullet U' \leq M' + \bullet U' \in [M'_0]_{N'}$, the multiset U' must be a set. As N' is unlabelled, this implies that the multiset $\ell'(U')$ is a set. Since $\ell(U) = \ell'(U')$, also $\ell(U)$, and hence U , must be a set. All its elements have the form execute_j^i for $i \leq^\# j \in T'$, since these are the only transitions in T with visible labels. Note that U' is completely determined by U , namely by $U' = \{i \mid \exists j. \text{execute}_j^i \in U\}$. We take

$$\begin{aligned} H_{M',U} &:= \sum_{p \in S'} (M' + \bullet U')(p) \cdot \{\text{distribute}_p\} \\ &\quad + \sum_{(M' + \bullet U')[j]} \left(\{\text{initialise}_j \cdot \text{fire}\} + \sum_{h <^\# j, \nexists \text{execute}_h^g \in U} \{\text{transfer}_j^h \cdot \text{fire}\} \right) \end{aligned}$$

Since N' is finitary, $H_{M',U} \in \mathbb{N}^{T+}$. Moreover, $\ell(H_{M',U}) \equiv \emptyset$.

Let $H \in NF$ with $M := M' + \bullet U' + (M_0 - M'_0) + \llbracket H \rrbracket - \bullet U \in \mathbb{N}^S$ and $M + \bullet U \in [M_0]_N$. Since $H \in NF$, and thus $\ell(H) \equiv \emptyset$, $H(\text{execute}_j^i) = 0$. From here on we apply Claim 1 and Claim 6 with $M + \bullet U$ and $M' + \bullet U'$ playing the rôles of M and M' . Note that the preconditions of these claims are met.

That $H(\text{execute}_j^i) = 0$ for all $i \leq^\# j \in T'$, together with (5.9) and the requirements (NF-1) and (NF-3) for normal forms, yields $H(t \cdot \text{elide}_i) \leq 0$ as well as $H(t \cdot \text{undo}_i) \leq 0$. Using this, (5.10)–(5.13) imply that

$$H(u) \leq 0 \quad \text{for each } u \in T_-. \quad (5.24)$$

Claim 7 Let $c \in T'$ and $p \in \bullet c$. Then

- if $H(\text{initialise}_c \cdot \text{fire}) > 0$ then $H(\text{fetch}_{i,j}^{p,c}) = 0$ for all $i \in p^\bullet$ and $j \geq^\# i$, and
- if $H(\text{transfer}_c^b \cdot \text{fire}) > 0$ for some $b <^\# c$ then $H(\text{fetch}_{i,j}^{p,c}) = 0$ for all $i \in p^\bullet$ and $j \geq^\# i$.

Proof: Suppose that $H(t \cdot \text{fire}) > 0$, for $t = \text{initialise}_c$ or $t = \text{transfer}_c^b$. Then (5.14) resp. (5.21) together with (5.24) implies that $H(t \cdot \text{reset}_\omega) = 0$ for each ω with $t \in \Omega_\omega$. In other words, $H(t \cdot \text{reset}_i) = 0$ for each $i \neq c$, so in particular for each $i \in p^\bullet$. Furthermore, $H(t \cdot \text{elide}_i) \geq 0$, by requirement (NF-3) of normal forms. With (5.10), this yields $\sum_{j \geq^\# i} H(\text{fetched}_j^i) \geq 0$, and (5.24) implies $H(\text{fetched}_j^i) = 0$ for each $j \geq^\# i$. Now (5.13, 5.24) gives $H(\text{fetch}_{i,j}^{p,c}) = 0$ for each $j \geq^\# i \in p^\bullet$. ■

We proceed to verify the requirements (5a)–(5g) of Theorem 9.

(5a) To show that $M_{M',U} \in \mathbb{N}^S$, it suffices to apply it to the preplaces of transitions in $H_{M',U} + U$, i. e.

$$\begin{aligned}
& \text{for all } p \in S' \\
& M_{M',U}(p) = 0 \\
& \text{for all } p \in S', j \in p^\bullet \\
& M_{M',U}(p_j) = \begin{cases} (M' + \bullet U')(p) - F'(p, j) & \text{if } (M' + \bullet U')[j] \\ (M' + \bullet U')(p) & \text{otherwise} \end{cases} \\
& \text{for all } j \in T' \\
& M_{M',U}(\pi_j) = \begin{cases} 0 & \text{if } (M' + \bullet U')[j] \\ 1 & \text{otherwise} \end{cases} \\
& \text{for all } j \leq^\# k \in T' \\
& M_{M',U}(\text{pre}_k^j) = \begin{cases} 1 & \text{if } (M' + \bullet U')[j] \wedge \text{execute}_k^j \notin U \\ -1 & \text{if } \neg(M' + \bullet U')[j] \wedge \text{execute}_k^j \in U \\ 0 & \text{otherwise} \end{cases} \\
& \text{for all } h <^\# j \in T' \\
& M_{M',U}(\pi_{h\#j}) = \begin{cases} 0 & \text{if } \exists \text{execute}_h^g \in U \vee (M' + \bullet U')[j] \\ 1 & \text{otherwise} \end{cases} \\
& \text{for all } h <^\# j \in T' \\
& M_{M',U}(\text{trans}_j^h\text{-in}) = \begin{cases} 1 & \text{if } (M' + \bullet U')[j] \wedge \exists \text{execute}_h^g \in U \\ 0 & \text{otherwise} \end{cases} \\
& \text{for all } h <^\# j \in T' \\
& M_{M',U}(\text{trans}_j^h\text{-out}) = \begin{cases} 1 & \text{if } (M' + \bullet U')[j] \wedge \nexists \text{execute}_h^g \in U \\ & \wedge \nexists \text{execute}_j^i \in U \\ -1 & \text{if } (\neg(M' + \bullet U')[j] \vee \exists \text{execute}_h^g \in U) \\ & \wedge \exists \text{execute}_j^i \in U \\ 0 & \text{otherwise} \end{cases}
\end{aligned}$$

For all these places s we indeed have that $M_{M',U}(s) \geq 0$, as the circumstances yielding the two exceptions above cannot occur:

- Suppose $\text{execute}_k^j \in U$ with $j \leq^\# k \in T'$. Then $j \in U'$, so $\bullet j \leq M' + \bullet U'$ and $(M' + \bullet U')[j]$. Consequently, $M_{M',U}(\text{pre}_k^j) \neq -1$ for all $j \leq^\# k \in T'$.
- Suppose $\text{execute}_j^i \in U$ with $i \leq^\# j \in T'$. Then $\bullet \text{execute}_j^i \leq \bullet U$, so $(M + \bullet U)[\text{execute}_j^i]$. Claim 6(e) with $M + \bullet U$ and $M' + \bullet U'$ in the rôles of M and M' yields $(M' + \bullet U')[j]$.

If moreover $\text{execute}_h^g \in U$ with $g \leq^\# h <^\# j$, then $\{g\} + \{i\} \leq U'$, so $\bullet\{g\} + \bullet\{i\} \leq M' + \bullet U'$ and $(M' + \bullet U')[\{g\} + \{i\}]$. In particular, $g \smile i$, and since N' is a structural conflict net, $\bullet g \cap \bullet i = \emptyset$. By Claim 6(e) – as above – $(M' + \bullet U')[h]$, so $\bullet g \cup \bullet h \cup \bullet j \cup \bullet i \leq M' + \bullet U' \in [M'_0]_{N'}$. Moreover, since $g \leq^\# h <^\# j \geq^\# i$, we have $\bullet g \cap \bullet h \neq \emptyset$, $\bullet h \cap \bullet i \neq \emptyset$ and $\bullet i \cap \bullet j \neq \emptyset$. Now in case also $\bullet h \cap \bullet i \neq \emptyset$, the transitions g , h and i constitute a fully reachable pure M; otherwise $h \smile i$ and h , j and i constitute a fully reachable pure M. Either way, we obtain a contradiction. Consequently, $M_{M',U}(\text{trans}_j^h\text{-out}) \neq -1$ for all $h <^\# j \in T'$.

- (5b) Suppose $M' \xrightarrow{a}$; say $M'[i]$ with $\ell(i) = a$. Let j be the largest transition in T' w.r.t. the well-ordering $<$ on T such that $i \leq^\# j$ and $(M' + \bullet U')[j]$. It suffices to show that $M_{M',U}[\text{execute}_j^i]$, i.e. that $M_{M',U}(\text{pre}_j^i) = 1$, $M_{M',U}(\text{trans}_j^h\text{-out}) = 1$ for all $h <^\# j$, and also $M_{M',U}(\pi_{j\#l}) = 1$ for all $l >^\# j$.

If $\text{execute}_j^i \in U$ we would have $i \in U'$ and hence $(M' + \bullet U')[2 \cdot \{i\}]$. Since N' is a finitary structural conflict net, this is impossible. Therefore $\text{execute}_j^i \notin U$ and, using the calculations from (a) above, also $M_{M',U}(\text{pre}_j^i) = 1$.

Let $h <^\# j$. To establish that $M_{M',U}(\text{trans}_j^h\text{-out}) = 1$ we need to show that there is no $k \leq^\# j$ with $\text{execute}_j^k \in U$ and no $g \leq^\# h$ with $\text{execute}_h^g \in U$. First suppose $\text{execute}_j^k \in U$ for some $k \leq^\# j$. Then $k \in U'$ and hence $(M' + \bullet U')[\{i\} + \{k\}]$. This implies $i \smile k$, and, as N' is a structural conflict net, $\bullet i \cap \bullet k = \emptyset$. Hence the transitions i , j and k are all different, with $\bullet i \cap \bullet j \neq \emptyset$ and $\bullet j \cap \bullet k \neq \emptyset$ but $\bullet i \cap \bullet k = \emptyset$. Moreover, the reachable marking $M' + \bullet U'$ enables all three of them. Hence N' contains a fully reachable pure M, which contradicts the assumptions of Theorem 11.

Next suppose $\text{execute}_h^g \in U$ for some $g \leq^\# h$. Then $(M' + \bullet U')[\text{execute}_h^g]$, hence $(M' + \bullet U')[h]$ by Claim 6(e). Moreover, $g \in U'$, hence $(M' + \bullet U')[\{i\} + \{g\}]$. This implies $g \smile i$, and $\bullet g \cap \bullet i = \emptyset$. Moreover, $\bullet g \cap \bullet h \neq \emptyset$, $\bullet h \cap \bullet j \neq \emptyset$ and $\bullet j \cap \bullet i \neq \emptyset$, while the reachable marking $M' + \bullet U'$ enables all these transitions. Depending on whether $\bullet h \cap \bullet i = \emptyset$, either h , j and i , or g , h and i constitute a fully reachable pure M, contradicting the assumptions of Theorem 11.

Let $l >^\# j$. To establish that $M_{M',U}(\pi_{j\#l}) = 1$ we need to show that there is no $k \leq^\# j$ with $\text{execute}_j^k \in U$ – already done above – and that $\neg(M' + \bullet U')[l]$. Suppose $(M' + \bullet U')[l]$. Considering that j was the largest transition with $i \leq^\# j$ and $(M' + \bullet U')[j]$, we cannot have $i <^\# l$. Hence the transitions i , j and l are all different, with $\bullet i \cap \bullet j \neq \emptyset$ and $\bullet j \cap \bullet l \neq \emptyset$ but $\bullet i \cap \bullet l = \emptyset$. Moreover, the reachable marking $M' + \bullet U'$ enables all three of them. Hence N' contains a fully reachable pure M, which contradicts the assumptions of Theorem 11.

- (5c) We have to show that $H(t) \leq H_{M',U}(t)$ for each $t \in T$.

- In case $t \in T_-$ this follows from (5.24) and $H_{M',U} \in \mathbb{N}^{T_+}$.
- In case $t = \text{execute}_j^i$ it follows since $\ell(H) \equiv \emptyset$.
- In case $t = \text{distribute}_p$ it follows from (5.20) and (5.24).
- Next let $t = \text{initialise}_c \cdot \text{fire}$ for some $c \in T'$. In case $H(\text{initialise}_c \cdot \text{fire}) \leq 0$ surely we have $H(\text{initialise}_c \cdot \text{fire}) \leq H_{M',U}(\text{initialise}_c \cdot \text{fire})$. So without limitation of generality we may assume that $H(\text{initialise}_c \cdot \text{fire}) > 0$. By (5.14, 5.24) we have $H(\text{initialise}_c \cdot \text{fire}) = 1$. Using (5.19), Claim 7, (5.24) and (5.20) we obtain, for all $p \in \bullet c$,

$$F'(p, c) \cdot H(\text{initialise}_c \cdot \text{fire}) \leq H(\text{distribute}_p) \leq (M' + \bullet U')(p).$$

Hence c is enabled under $M' + \bullet U'$, which implies $H_{M',U}(\text{initialise}_c \cdot \text{fire}) = 1$.

- Let $t = \text{transfer}_c^b \cdot \text{fire}$ for some $b < \#_c \in T'$. As above, we may assume $H(\text{transfer}_c^b \cdot \text{fire}) > 0$. By (5.21, 5.24) we have $H(\text{transfer}_c^b \cdot \text{fire}) = 1$. Using (5.24) and that $H(\text{execute}_b^g) = 0$ for all $g \leq \# b$, it follows that $(M + \bullet U)(\pi_{b\#c}) = 0$. Hence $\neg(M + \bullet U)[\text{execute}_b^g]$ for all $g \leq \# b$, and thus $\nexists \text{execute}_b^g \in U$. For all $p \in \bullet c$ we derive

$$\begin{aligned}
& F'(p, c) \cdot H(\text{transfer}_c^b \cdot \text{fire}) \\
& \leq F'(p, c) \cdot (H(\text{transfer}_c^b \cdot \text{fire}) - H(\text{transfer}_c^b \cdot \text{undone})) \quad (5.24) \\
& \leq F'(p, c) \cdot (H(\text{initialise}_c \cdot \text{fire}) - H(\text{initialise}_c \cdot \text{undo}(\text{trans}_c^b\text{-in}))) \quad (5.15) \\
& \leq F'(p, c) \cdot (H(\text{initialise}_c \cdot \text{fire}) - H(\text{initialise}_c \cdot \text{undone})) \quad (5.12) \\
& = [\text{the same as above}] + \sum_{j \geq \# i \in p \bullet} F'(p, i) \cdot H(\text{fetch}_{i,j}^{p,c}) \quad (\text{Claim 7}) \\
& \leq H(\text{distribute}_p) \quad (5.19) \\
& \leq (M' + \bullet U')(p) + \sum_{\{i \in T' \mid p \in i \bullet\}} H(\text{finalise}_i) \quad (5.20) \\
& \leq (M' + \bullet U')(p) \quad (5.24).
\end{aligned}$$

Hence $(M' + \bullet U')[c]$, and thus $H_{M',U}(\text{transfer}_c^b) = 1$.

(5d) If $u \notin T_-$, yet $H(u) \neq 0$, then u is either distribute_p , $\text{initialise}_j \cdot \text{fire}$ or $\text{transfer}_j^h \cdot \text{fire}$ for suitable $p \in S'$ or $h, j \in T'$. For $u = \text{distribute}_p$ the requirement follows from Claim 6(c); otherwise Property (NF-2), together with (5.12), guarantees that $H(u) \geq 0$.

(5e) If $H(t) > 0$ and $H(u) < 0$, then $t \in T_+$ and $u \in T_-$. The only candidates for $\bullet t \cap \bullet u \neq \emptyset$ are

- $p_c \in \bullet(\text{initialise}_c \cdot \text{fire}) \cap \bullet(\text{fetch}_{i,j}^{p,c})$ for $p \in S'$, $c, i \in p \bullet$ and $j \geq \# i$,
- $\text{trans}_c^b\text{-in} \in \bullet(\text{transfer}_c^b \cdot \text{fire}) \cap \bullet(\text{initialise}_c \cdot \text{undo}(\text{trans}_c^b\text{-in}))$ for $b \leq \# c \in T'$.

We investigate these possibilities one by one.

- $H(\text{initialise}_c \cdot \text{fire}) > 0 \wedge H(\text{fetch}_{i,j}^{p,c}) < 0$ cannot occur by Claim 7.
 - Suppose $H(\text{transfer}_c^b \cdot \text{fire}) > 0$. By (5.21, 5.24) we have $H(\text{transfer}_c^b \cdot \text{fire}) = 1$. Through the derivation above, in the proof of requirement (c), using (5.24, 5.15, 5.12), Claim 7 and (5.19), we obtain $H(\text{distribute}_p) \geq F'(p, c)$ for all $p \in \bullet c$. Now Claim 6(d) yields $H(\text{finalise}_i) = 0$ for all $i \not\equiv c$. By (5.10) and (5.24) we obtain $H(\text{initialise}_c \cdot \text{reset}_i) = 0$ for each such i . Hence $\sum_{i \not\equiv c} H(\text{initialise}_c \cdot \text{reset}_i) = 0$, and $H(\text{initialise}_c \cdot \text{undo}(\text{trans}_c^b \cdot \text{in})) = 0$ by (5.12, 5.24).
- (5f) If $H(u) < 0$ and $(M + \bullet U)[t]$ with $\ell(t) \neq \tau$, then $t = \text{execute}_j^i$ for some $i \leq^\# j \in T'$ and $u \in T_-$. The only candidates for $\bullet t \cap \bullet u \neq \emptyset$ are
- $\text{pre}_j^i \in \bullet(\text{execute}_j^i) \cap \bullet(\text{initialise}_j \cdot \text{undo}(\text{pre}_j^i))$ and
 - $\text{trans}_j^h \cdot \text{out} \in \bullet(\text{execute}_j^i) \cap \bullet(\text{transfer}_j^h \cdot \text{undo}(\text{trans}_j^h \cdot \text{out}))$ for $h <^\# j$.

We investigate these possibilities one by one.

- Suppose $(M + \bullet U)[\text{execute}_j^i]$. By Claim 6(b), $H(\text{finalise}^k) \geq 0$ for each $k \not\equiv i$. By (5.10) and (5.24) we obtain $H(\text{initialise}_i \cdot \text{reset}_k) = 0$ for each such k . Hence $\sum_{k \not\equiv i} H(\text{initialise}_i \cdot \text{reset}_k) = 0$, and thus $H(\text{initialise}_i \cdot \text{undo}(\text{pre}_j^i)) = 0$ by (5.12, 5.24).
 - Suppose $(M + \bullet U)[\text{execute}_j^i]$ and $h <^\# j$. By Claim 6(b), then $H(\text{finalise}^k) \geq 0$ for each $k \not\equiv j$. By (5.10) and (5.24) then follows $H(\text{transfer}_j^h \cdot \text{reset}_k) = 0$ for each such k . So $\sum_{k \not\equiv j} H(\text{transfer}_j^h \cdot \text{reset}_k) = 0$, and $H(\text{transfer}_j^h \cdot \text{undo}(\text{trans}_j^h \cdot \text{out})) = 0$ by (5.12, 5.24).
- (5g) Suppose $(M + \bullet U)[\{t\} + \{u\}]_N$, and $i, k \in T'$ with $\ell'(i) = \ell(t)$ and $\ell'(k) = \ell(u)$. Since the net N' is unlabelled, t and u must have the form execute_j^i and execute_l^k for some $j >^\# i$ and $l >^\# k$. Claim 4 yields $\neg(i \not\equiv k)$ and hence $\bullet i \cap \bullet k = \emptyset$. \square

Thus, we have established that the conflict replicating implementation $\mathcal{I}(N')$ of a finitary unlabelled structural conflict net N' without a fully reachable pure M is branching ST-bisimilar with explicit divergence to N' . It remains to be shown that $\mathcal{I}(N')$ is essentially distributed.

Lemma 19 Let N be the conflict replicating implementation of a finitary net $N' = (S', T', F', M'_0, \ell')$; let $j, l \in T'$, with $l >^\# j$. Then no two transitions from the set

$$\{\text{execute}_j^i \mid i \leq^\# j\} \cup \{\text{transfer}_l^j \cdot \text{fire}\} \cup \{\text{transfer}_l^j \cdot \text{undo}(\text{trans}_l^j \cdot \text{out})\} \\ \cup \{\text{execute}_l^k \mid k \leq^\# l\}$$

can fire concurrently.

Proof: For each $i \leq^\# j$ pick an arbitrary preplace q_i of i . The set

$$\{\text{fetch}_{i,j}^{q_i,i} \cdot \text{in}, \text{fetch}_{i,j}^{q_i,i} \cdot \text{out} \mid i \leq^\# j\} \\ \cup \{\pi_{j \# l}, \text{trans}_l^j \cdot \text{out}, \text{took}(\text{trans}_l^j \cdot \text{out}, \text{transfer}_l^j), \rho(\text{transfer}_l^j)\}$$

is an *S-invariant*: there is always exactly one token in this set. This is the case because there is exactly one token initially (on $\pi_{j\#l}$) and each transition from N has as many (with multiplicities) preplaces as postplaces in this set. The transitions from

$$\{\text{execute}_j^i \mid i \leq \#j\} \cup \{\text{transfer}_l^j \cdot \text{fire}\} \cup \{\text{transfer}_l^j \cdot \text{undo}(\text{trans}_l^j\text{-out})\} \\ \cup \{\text{execute}_l^k \mid k \leq \#l\}$$

each have a preplace in this set. Hence no two of them can fire concurrently. \square

Lemma 20 Let N be the conflict replicating implementation $\mathcal{I}(N')$ of a finitary unlabelled structural conflict net $N' = (S', T', F', M'_0, \ell')$ without a fully reachable pure M . Then for any $i \leq \#j \stackrel{\#}{=} c \in T'$ and $f \in (\text{initialise}_c)^{far}$, the transitions execute_j^i and $\text{initialise}_c \cdot \text{undo}(f)$ cannot fire concurrently.

Proof: Suppose these transitions can fire concurrently, say from the marking $M \in [M_0]_N$. By Claim 3, there are $M' \in [M'_0]_{N'}$ and $G \in \mathbb{Z}^T$ such that (B)–(N) hold. Let $t := \text{initialise}_c$, $G_1 := G + \{t \cdot \text{undo}(f)\}$ and $M_1 := M + \llbracket t \cdot \text{undo}(f) \rrbracket$. Then (5.12), applied to the triples (M, M', G) and (M_1, M', G_1) , yields

$$\sum_{\{\omega \mid t \in \Omega_\omega\}} G(t \cdot \text{reset}_\omega) \leq G(t \cdot \text{undo}(f)) < G_1(t \cdot \text{undo}(f)) \leq \sum_{\{\omega \mid t \in \Omega_\omega\}} G_1(t \cdot \text{undo}_\omega) \\ = \sum_{\{\omega \mid t \in \Omega_\omega\}} G(t \cdot \text{undo}_\omega).$$

Hence, there is an ω with $t \in \Omega_\omega$ and $G(t \cdot \text{reset}_\omega) < G(t \cdot \text{undo}_\omega)$. This ω must have the form $k \in T'$ with $k \stackrel{\#}{=} c$. We now obtain

$$\begin{aligned} 0 &= G(\text{finalise}_k) && \text{(by (C))} \\ &\leq G(t \cdot \text{elide}_k) + G(t \cdot \text{reset}_k) && \text{(by (5.10))} \\ &< G(t \cdot \text{elide}_k) + G(t \cdot \text{undo}_k) \\ &\leq \sum_{l \geq \#k} G(\text{execute}_l^k) && \text{(by (5.9)).} \end{aligned}$$

Hence, there is an $l \geq \#k \stackrel{\#}{=} c$ with $G(\text{execute}_l^k) > 0$. By (M) we obtain $\neg(j \stackrel{\#}{=} k)$, so $\bullet j \cap \bullet k = \emptyset$. Additionally, we have $\bullet j \cap \bullet c \neq \emptyset$ and $\bullet c \cap \bullet k \neq \emptyset$. By (N) we obtain $M'[j]$, and by (D) and (E) $M'[k]$. Furthermore, by (5.12), $G(t \cdot \text{undo}(f)) < G_1(t \cdot \text{undo}(f)) \leq G_1(t \cdot \text{fire}) = G(t \cdot \text{fire})$, so, for all $p \in \bullet c$,

$$\begin{aligned} F'(p, c) &\leq F'(p, c) \cdot (G(t \cdot \text{fire}) - G(t \cdot \text{undo}(f))) \\ &\leq F'(p, c) \cdot (G(t \cdot \text{fire}) - G(t \cdot \text{undone})) && \text{(by (5.12))} \\ &\leq G(\text{distribute}_p) - \sum_{j \geq \#i \in p \bullet} F'(p, i) \cdot G(\text{fetch}_{i,j}^{p,c}) && \text{(by (5.19))} \\ &\leq G(\text{distribute}_p) && \text{(by (E) and (5.13))} \\ &\leq M'(p) && \text{(by (D)).} \end{aligned}$$

It follows that $M'[c]$. Thus N' contains a fully reachable pure M , which contradicts the assumptions of Lemma 20. \square

Theorem 12 Let N be the conflict replicating implementation $\mathcal{I}(N')$ of a finitary unlabelled structural conflict net N' without a fully reachable pure \mathbf{M} . Then N is essentially distributed.

Proof: We take the canonical distribution D of N , in which \equiv_D is the equivalence relation on places and transitions generated by Condition (1) of Definition 33. We need to show that this distribution satisfies Condition (2') of Definition 34. A given transition t with $\ell(t) \neq \tau$ must have the form execute_j^i for some $i \leq^\# j \in T'$. By following the flow relation of N one finds the places and transitions that, under the canonical distribution, are co-located with execute_j^i :

$$\begin{array}{ccc}
 \pi_{j\#l} \rightarrow \text{transfer}_l^j \cdot \text{fire} \leftarrow \text{trans}_l^j\text{-in} \rightarrow \text{initialise}_l \cdot \text{undo}(\text{trans}_l^j\text{-in}) & & \\
 \downarrow & & \uparrow \\
 \text{execute}_j^i & & \text{take}(\text{trans}_l^j\text{-in}, \text{initialise}_l) \\
 \uparrow & & \\
 \text{trans}_j^h\text{-out} \rightarrow \text{transfer}_j^h \cdot \text{undo}(\text{trans}_j^h\text{-out}) \leftarrow \text{take}(\text{trans}_j^h\text{-out}, \text{transfer}_j^h) & & \\
 \downarrow & & \\
 \text{execute}_j^g & & \\
 \uparrow & & \\
 \text{pre}_j^g \rightarrow \text{initialise}_g \cdot \text{undo}(\text{pre}_j^g) \leftarrow \text{take}(\text{pre}_j^g, \text{initialise}_g) & &
 \end{array}$$

for all $l \geq^\# j$, $h <^\# j$ and $g \leq^\# j$. Note that the chain starting at $\text{transfer}_l^j \cdot \text{fire}$ only exists for $l >^\# j$. We need to show that none of these transitions can happen concurrently with execute_j^i . For transitions $\text{transfer}_l^j \cdot \text{fire}$ and execute_j^g this follows directly from Lemma 19. For $\text{transfer}_j^h \cdot \text{undo}(\text{trans}_j^h\text{-out})$ this also follows from Lemma 19, in which j , k and l play the rôle of the current h , i and j . For the transitions $\text{initialise}_l \cdot \text{undo}(\text{trans}_l^j\text{-in})$ and $\text{initialise}_g \cdot \text{undo}(\text{pre}_j^g)$ this has been established in Lemma 20. \square

Our main result follows by combining Theorems 11, 12 and 3:

Theorem 13 Let N be a finitary unlabelled structural conflict net without a fully reachable visible pure \mathbf{M} . Then N is distributable up to \approx_{bSTb}^Δ . \square

Corollary 5 Let N be a finitary unlabelled structural conflict net. Then N is distributable iff it has no fully reachable visible pure \mathbf{M} . \square

Chapter 6

M-Containing Nets

6.1 Instability of Ms

Certainly, there are behavioural equivalences for which Ms are not necessarily preserved by equivalent implementations, e.g. the trivial equivalence which relates all nets.

A more insightful statement was shown in my diploma thesis:

Theorem 14 Let N be a finite, unlabelled net. There exists a finite, distributed net N' which is weak completed step trace equivalent to N .

Proof: Take N' to be the N'' of Theorem 5.2.1 of [Sch08]. □

The proof given in [Sch08] involves a concrete construction and comes with an invariant property of similar flavour as the one given in Chapter 5 – but was written before I invented the $M \oplus H$ resp. $\llbracket H \rrbracket$ notation. I omit it here as a service to the reader and also because it was published as a thesis already.

This bound of instability is not exact, but covers a relevant part of the linear time spectrum. Nonetheless, the stability of Ms between weak completed step trace and step failures equivalence is still open.

6.2 Stability of Ms

In Theorem 8 we have already shown – as a kind of motivation – that Ms cannot be eliminated by step failure equivalent implementations. Hence Ms are stable for any finer equivalence relations as well.

In what follows a similar stability result is achieved for a linear time equivalence, completed pomset trace equivalence, where instead of the branching structure of the system, the causal relationship between events is considered. The material was published in [SPG11].

As completed pomset trace equivalence is a very linear-time equivalence, it disregards the decision structure of a system and an implementation like the

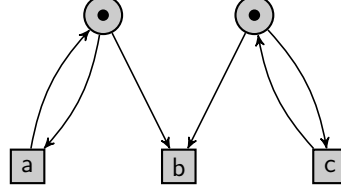


Figure 6.1: A repeated pure **M**. A finite, 1-safe, undistributable net used as a counterexample.

one of Figure 6.2, which simply provides a separate branch for each possible maximal process of the original net, would be fully satisfactory. In practice though, such an infinite implementation is unwieldy to say the least. If however infinite implementations are ruled out, the main result of this section shows that no valid implementation of the repeated pure **M** of Figure 6.1 exists.

Before we consider the first theorem of this section, let us concentrate on two auxiliary lemmata. The first states that the careful introduction of a τ -transition before an arbitrary transition of a net, as described below, does not significantly influence the properties of that net.

Lemma 21 Let $N = (S, T, F, M_0, \ell)$ be a finite, 1-safe, distributed net with the distribution function D . Let $t \in T$. The net $N' = (S', T', F', M_0, \ell')$ with

- $S' = S \cup \{s_t\}$,
- $T' = T \cup \{\tau_t\}$,
- $F'(x, y) = \begin{cases} F(x, t) & \text{iff } x \in S, y = \tau_t, \\ 1 & \text{iff } x = \tau_t, y = s_t, \\ 1 & \text{iff } x = s_t, y = t, \\ F(x, y) & \text{otherwise, and} \end{cases}$
- $\ell'(x) = \begin{cases} \tau & \text{if } x = \tau_t \\ \ell(x) & \text{otherwise} \end{cases}$

is finite, 1-safe, distributed and completed pomset trace equivalent to N .

Proof: (Sketch)

N' is finite as only two new elements were introduced. N' is completed pomset trace equivalent to N . Given a process (\mathcal{N}, π) of N , a process of N' can be constructed by refining in \mathcal{N} every transition u in the same manner as $\pi(u)$ was in N . For the reverse direction, note that in every maximal processes of N' , $\pi(u) = t \implies \pi(\bullet u) = \{s_t\} \wedge \pi(\bullet s_t) = \{\tau_t\}$. By fusing u , $\bullet u$, and $\bullet \bullet u$ into a single transition v whenever $\pi(u) = t$ and setting the process mapping of v to t , a maximal process of N' can be transformed into a maximal process of N . For the same reason, N' is also 1-safe.

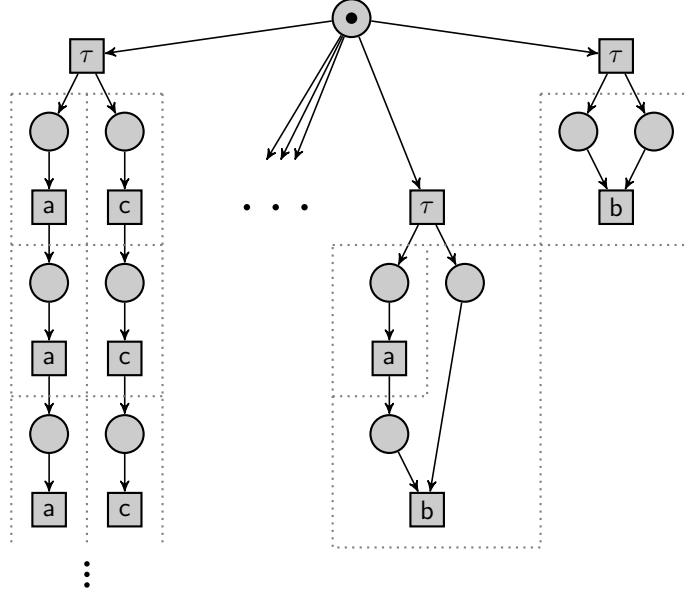


Figure 6.2: An infinite implementation of Figure 6.1, constructed by taking every maximal process and initially choosing one, location borders dotted.

N' is distributed with the distribution function

$$D'(x) := \begin{cases} D(t) & \text{if } x = s_t \vee x = \tau_t \\ D(x) & \text{otherwise} \end{cases}.$$

The places in $\bullet\tau_t$ are on $D(t) = D'(\tau_t)$. $D'(s_t) = D(t) = D'(t)$. Hence all transitions are on the same location as their preplaces. No new parallelism is introduced, as a parallel firing of either τ_t or t with some other transition u can only occur if t and u could already fire in parallel in N . \square

For now we only consider 1-safe nets. Formally, we restrict ourselves to *contact-free nets*, where in every reachable dependency marking $M_1 \in [M_0]^d$ for all $t \in T$ with $\bullet t \subseteq \text{pr}_1(M_1)$

$$(\text{pr}_1(M_1) \setminus \bullet t) \cap t^\bullet = \emptyset.$$

For such nets, in Definition 24 we can just as well consider a transition t to be enabled in M iff $\bullet t \subseteq \text{pr}_1(M)$, and two transitions to be independent when $\bullet t \cap \bullet u = \emptyset$.

The later proof that Figure 6.1 is non-implementable depends crucially on this 1-safety assumption. We conjecture however, that the result itself will hold, even if non-safe implementations are allowed.

Next we show, that if a dependency marking is reached twice during an execution, the dependencies of all tokens consumed and produced by a transition firing in such a cycle are equal.

Lemma 22 Let $N = (S, T, F, M_0, \ell)$ be a finite, 1-safe net. Moreover let $t_s, t_{s+1}, \dots, t_{e-1}, t_e \in T$ be a sequence of transitions leading from a reachable dependency marking M_{base} to the same, i. e. $M_{base}[\{t_s\}]_N^d \cdots [\{t_e\}]_N^d M_{base}$. Then every t_i produced tokens that were dependent on the same labels as the tokens on its preplaces.

Proof: Assume the opposite, i. e. there is a t_i for $s \leq i \leq e$ such that t_i consumed an L -independent token from one of its preplaces (for some $L \subseteq \text{Act}$), but produced no L -independent tokens. This L -independent token needs to be replaced to again reach M_{base} . However the replacement token needs to be L -independent as otherwise a dependency marking different from M_{base} would be reached. This token can thus not depend on any of the tokens produced by t_i , as it would then not be L -independent. In other words, had t_i not fired, a new L -independent token could also have been produced on its preplaces, i. e. N would not be 1-safe, violating the assumptions. Hence no such t_i can be fired, or equivalently, every t_i produced tokens that were dependent on the same labels as the tokens on its preplaces (which hence all have the same dependencies). \square

We will now show that, given an arbitrary finite, 1-safe net, it is not possible in general to find a finite, 1-safe, and distributed net which is completed pomset trace equivalent to the original. As a counterexample, consider the repeated pure **M** of Figure 6.1. It is a simple net allowing to perform several transitions of a and c in parallel, and terminating with a single transition b . The main argument of the following proof proceeds as follows: To perform an arbitrary number of a and c -transitions within a finite net there has to be a loop. To terminate with b the process has to escape from that loop by disabling all transitions leading to a or c . Therefore either a single token is consumed that is dependent on a as well as on c , or two different tokens – one a -dependent and one c -dependent – are consumed. In the first case an additional iteration of the loop results in an additional causal dependency, i. e. in a causal dependency between a and c . In the second case the net is not distributed in the sense of Definition 33.

Theorem 15 It is in general impossible to find for a finite, 1-safe net a distributed, completed pomset trace equivalent, finite, 1-safe net.

Proof: Via the counterexample given in Figure 6.1. Suppose a finite, 1-safe, distributed net N_{impl} , which is completed pomset trace equivalent to the net of Figure 6.1, would exist. By refining every b -labelled transition in N_{impl} into two transitions in the manner of Lemma 21, a new net $N = (S, T, F, M_0, \ell)$ is derived. By Lemma 21 this new net is finite, 1-safe, distributed and completed pomset trace equivalent to the net in Figure 6.1 since N_{impl} is.

N has $|S|$ places and 3 different labels, every place can hold either no token, or a token dependent on any possible combination of the three labels. Since N is

finite so is $|S|$. Hence N has at most $(1+2^3)^{|S|}$ reachable dependency markings. Let $m := 9^{|S|}$. N is able to fire $(ac)^mb$ without any step containing more than a single transition since the net of Figure 6.1 is and the two are assumed to be completed pomset trace equivalent. Let G_1, G_2, \dots, G_n be the steps fired while doing so. $|G_i| = 1$ for all i . In the course of firing that sequence, at least one dependency marking is bound to be reached twice. Of all those dependency markings which occur twice, we take the one occurring last while firing $(ac)^mb$ and call it M_{base} . Let $G_s, G_{s+1}, \dots, G_{e-1}, G_e$ be a sequence of steps between two occurrences of M_{base} , i.e. $M_0 \times \{\emptyset\} [G_1]^d [G_2]^d \dots M_{base} [G_s]^d \dots [G_e]^d M_{base} \dots [G_n]^d$.

Using Lemma 22 the transitions of the steps G_s to G_e can be partitioned into subsets T_X based on the dependencies of the tokens they produced and consumed. A set T_X includes all transitions producing X -dependent, $(\text{Act} \setminus X)$ -independent tokens. By firing

$$G_s \cap T_{\{a\}}, G_{s+1} \cap T_{\{a\}}, \dots, G_e \cap T_{\{a\}} \text{ (skipping empty steps)}$$

repeatedly, $M_{base} \xRightarrow{a^m} d$. By firing

$$G_s \cap T_{\{c\}}, G_{s+1} \cap T_{\{c\}}, \dots, G_e \cap T_{\{c\}} \text{ (skipping empty steps)}$$

repeatedly, $M_{base} \xRightarrow{c^m} d$. We now search for the marking, where the decision to fire b is made.

Assume a reachable dependency marking M'' of N with $M'' \xRightarrow{a^m} d$. If $M'' \not\xRightarrow{c^m} d$ then $M''' \not\xRightarrow{c^m} d$ for all M''' reachable from M'' since c cannot be enabled using tokens produced by a transition labelled a or b . Otherwise there would exist a pomset of N in which a c is causally dependent on an a or b . Such a pomset however does not exist for the net of Figure 6.1 thereby violating the assumption of completed pomset trace equivalence. If however c is not re-enabled after M'' a maximal process including finitely many c but infinitely many a 's can be produced also leading to a pomset not present in the net of Figure 6.1. The same argument can be applied with the rôles of a and c reversed, hence $M'' \xRightarrow{a^m} d$ iff $M'' \xRightarrow{c^m} d$.

We start from M_{base} and start to fire the steps G_s, G_{s+1}, \dots, G_n until a^m cannot be fired any more for the first time. This step always exists as after b no further a 's or c 's may be fired. Call the single transition in that step t_b . The marking right before that transition fired, we call M , the one right after it M' . Not only $M \xRightarrow{a^m} d$ but also $M \xRightarrow{c^m} d$ and not only $M' \xRightarrow{a^m} d$ but also $M' \xRightarrow{c^m} d$, as both M and M' are reachable markings.

t_b is not itself labelled b , as the refined net has a τ -transition before the b , and once a token resides on the intermediate place, no a -transitions can be fired any more, as otherwise a pomset where an a which is not a causal predecessor to a b would be produced, again not existing for the net of Figure 6.1.

To disable the trace a^m , the transition t_b needed to consume a token. If t_b had not fired, some $G_i \cap T_{\{a\}}$, $s \leq i \leq e$ could have consumed that token, hence that token must be a -dependent, c -independent. Similarly, t_b must have consumed a token which could have led to c^m . This token needs to be c -dependent, a -independent. Hence t_b has at least two preplaces, which in turn

are also preplaces to two different transitions, call them t_a and t_c , which then lead to a^m and c^m respectively.¹ As they have common preplaces t_a , t_b and t_c are on the same location.

From M the net can fire a^m consuming only a -dependent, c -independent tokens. It can also fire c^m consuming only c -dependent, a -independent tokens.

Hence there is a sequence of steps leading from M to a marking where t_a is enabled, yet only a -dependent, c -independent tokens have been removed or added. Similarly there is a firing sequence leading from M to a marking where t_c is enabled, yet only c -dependent, a -independent tokens have been removed or added. As they change disjoint sets of tokens, these two firing sequences can be concatenated, thereby leading to a marking where t_a and t_c are concurrently enabled, yet they are on the same location, thereby violating the implementation requirement that N is distributed. \square

Note that the self-loops of the counterexample are not critical to the success of the proof but any larger subnet returning the tokens would suffice.

This section only considered 1-safe nets as possible implementations. We conjecture however, that the proof of Theorem 15 can be extended to non-safe nets as well, as from a place where tokens of different dependency mix, a transition can always choose the most-dependent token. In particular a transition intended to produce independent tokens cannot have such a place as a preplace. Hence every part of the net providing independent tokens can do so without depending on firings of labelled transitions. The number of independent tokens produced on a place where a labelled transition consumes them is thus either finite over every run of the system, or unbounded even without any labelled transition ever firing. In both cases that place is unsuitable for disabling a potentially infinitely often occurring loop. If only finitely many tokens are produced, the loop can no longer happen infinitely often, if an unbounded number of tokens can be produced, no disabling can be guaranteed.

6.3 Infinite Number of Less Restrictive Classes

So far, we have identified partially and fully reachable Ns and fully reachable pure Ms as crucial structures when considering distributed system implementations. Each of them is in a particular sense not distributable, and both structures are stable with respect to branching time semantics, hence cannot be removed without changing behaviour. As² every fully reachable pure M also contains a partially and fully reachable N, these two structures, respectively the net classes defined by their absence, form a hierarchy of distributability. This hierarchy can easily be extended in the direction of more distributability by disallowing all conflicts, generating an even smaller, but easily implementable class of nets.

¹The removal of the token leading to a^m and the one leading to c^m must indeed be done by a single transition t_b as only a single transition was fired between M and M' and both traces were possible in M but impossible in M' .

²in self-loop free nets, to be exact

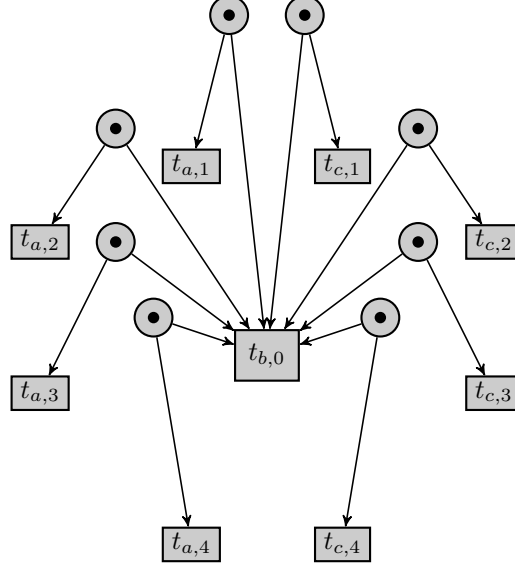


Figure 6.3: The arthropod net of degree 4.

The other direction however is not so clear. If by some means, Ms would be implementable, would this allow the implementation of all nets in a distributed fashion?

Assume a way was found to implement a prior impossible structure and waive – for sets of transitions of size n – the requirement that two transitions need to be co-located if they had a conflict. If these sets would be allowed to overlap, the answer would be trivially positive even for $n = 2$ by putting each pair of transitions into such a set. However, this would not model *connecting* some atomic building blocks implementing a given structure distributedly, as the same transition would find itself included in multiple blocks. To model connected atomic building blocks, such sets of assumed-to-be distributed transitions would need to be disjoint.

Definition 50 Let $N = (S, T, F, M_0, \ell)$ be a net. It is *distributed modulo atomic building blocks of size n* iff there exists a set Block of blocks and a block assignment function $B : T \rightarrow \text{Block} \cup \{\perp\}$ with $\perp \notin \text{Block}$ and $\forall b \in \text{Block}. n \geq |\{t \mid B(t) = b\}|$ such that there exists a location function $D : S \cup T \rightarrow \text{Loc}$ such that $\forall t, u \in T. (\bullet t \cap \bullet u \neq \emptyset \Rightarrow D(t) = D(u) \vee B(t) = B(u)) \wedge (t \sim u \Rightarrow D(t) \neq D(u))$.

Is there an n such that all nets are distributable modulo atomic building blocks of size n ?

Definition 51 Let $n \in \mathbb{N}$. The *arthropod net of degree n* (Figure 6.3 explains³ the name) is the net $N = (S, T, F, M_0, \ell)$ defined as

$$\begin{aligned} S &= \{s_{a,i}, s_{c,i} \mid 1 \leq i \leq n\} \\ T &= \{t_{b,0}\} \cup \{t_{a,i}, t_{c,i} \mid 1 \leq i \leq n\} \\ F(s_{x,i}, t_{y,j}) &= \begin{cases} 1 & \text{if } y = b \vee y = x \\ 0 & \text{otherwise} \end{cases} \\ F(t_{y,j}, s_{x,i}) &= 0 \\ M_0 &= S \\ \ell &= \text{Id}_T \end{aligned}$$

Proposition 6 The arthropod net of degree n (ArN_n) is not distributed modulo atomic building blocks of size n .

Proof: For each $1 \leq i \leq n$ we have that $t_{a,i} \smile t_{c,i}$, $\bullet t_{a,i} \cap \bullet t_{b,0} \neq \emptyset$, and $\bullet t_{b,0} \cap \bullet t_{c,i} \neq \emptyset$. Thus for each i , we have $D(t_{a,i}) \neq D(t_{c,i})$ and hence $B(t_{a,i}) = B(t_{b,0})$ or $B(t_{b,0}) = B(t_{c,i})$. It follows that $|\{t \mid B(t) = B(t_{b,0})\}| \geq n + 1$. \square

Theorem 16 If N is step failures equivalent to ArN_n then N is not distributed modulo atomic building blocks of size n .

Proof: Consider any i with $1 \leq i \leq n$. $M_0 \xrightarrow{\{t_{a,i}, t_{c,i}\}} \cdot$ and $M_0 \xrightarrow{\{t_{b,0}\}} \cdot$. Moreover $M_0 \xrightarrow{\{t_{a,i}, t_{b,0}\}} \cdot$ and $M_0 \xrightarrow{\{t_{b,0}, t_{c,i}\}} \cdot$. Finally also $M_0 \not\xrightarrow{\tau}$. Hence

$$\langle \varepsilon, \{\{t_{a,i}, t_{c,i}\}\} \rangle, \langle \varepsilon, \{\{t_{b,0}\}\} \rangle \notin \mathcal{F}(ArN_n) = \mathcal{F}(N)$$

and

$$\langle \varepsilon, \{\{t_{a,i}, t_{b,0}\}, \{t_{b,0}, t_{c,i}\}\} \rangle \in \mathcal{F}(ArN_n) = \mathcal{F}(N).$$

Converting from failure pairs back to transitions in N , we find for each $1 \leq i \leq n$ that $\langle \varepsilon, \{\{t_{a,i}, t_{c,i}\}\} \rangle \notin \mathcal{F}(N)$ hence there must exist at least one transition labelled $t_{a,i}$ which is initially enabled and we shall call $u_{a,i}$. From the labels $t_{c,i}$ we likewise obtain enabled transitions $u_{c,i}$. As the step containing both these transitions is not in the failure set, they can fire in the same step, i.e. $u_{a,i} \smile u_{c,i}$.

From $\langle \varepsilon, \{\{t_{b,0}\}\} \rangle \notin \mathcal{F}(N)$ we obtain an enabled transition $u_{b,0}$ labelled $t_{b,0}$.

As $\langle \varepsilon, \{\{t_{a,i}, t_{b,0}\}\} \rangle \in \mathcal{F}(N)$, it follows that $u_{a,i}$ and $u_{b,0}$ must not fire in the same step. As both are enabled after the empty trace ε , there must be a preplace shared between them, i.e. $\bullet u_{a,i} \cap \bullet u_{b,0} \neq \emptyset$. The same argument can be repeated for $u_{c,i}$ giving $\bullet u_{c,i} \cap \bullet u_{b,0} \neq \emptyset$.

This essentially replicated ArN_n into the new net, in particular for each i we have $D(u_{a,i}) \neq D(u_{c,i})$ and with $u_{a,i} \smile u_{c,i}$ then $B(u_{a,i}) = B(u_{b,0})$ or $B(u_{b,0}) = B(u_{c,i})$. Hence $|\{u \mid B(u) = B(u_{b,0})\}| \geq n + 1$ and N is not distributed modulo atomic building blocks of size n . \square

³The obvious name for $n = 3$ would be *insect nets*, for $n = 4$ *spider nets*.

The above proof is basically a replicated application of Lemma 12 and Lemma 13, which in their original form however do not yield the necessary identification of the $u_{b,0}$ transition across different i , hence the above, longer proof.

From Theorem 16 follows that the classes of nets distributable with respect to step failures equivalence modulo atomic building blocks of size n are strictly larger for increasing $n \geq 2$. The hierarchy of distributability can thus be extended towards synchronous nets infinitely.

If one so desires, this classification can be made finer and non-linear by not allowing arbitrary building blocks of size n , but only certain structures. Of course one would need to show stability of each of these structures with respect to step failures equivalence to establish that they indeed induce a new net class, but the proof method of Theorem 16 can be applied to a variety of structures.

Nonetheless, the above already suffices to shatter the hope of creating arbitrary distributed systems after having found a synchronisation mechanism for n transitions.

Chapter 7

Distributed Net Implementation

After many theorems have been presented about nets, it would be nice to test whether these insights can be applied practically. Also, in the spirit of Knuth’s “Beware of bugs in the above code; I have only proved it correct, not tried it” [Knu] the complex proof of Section 5.3 should be followed by actually trying it out. To this end, (a prototype of) a compiler was developed, which takes a distributed net and compiles its locations to Linux binaries which connect to each other using TCP/IP networking and actually execute the net in a distributed fashion. The electronic version of this thesis contains the compiler sources as a pdf-attachment to this page.

Within this framework, the constructions presented in Section 5.1.3 and Figure 5.1 have been implemented to increase the degree of distribution of a net. This allows to experimentally measure the concrete overhead of these transformations in practice.

As feared, these experiments also highlighted an oversight in the construction given in [GGSU13] which was corrected during preparation of this thesis: Where we originally had $l >^{\#} j$ it should have said $l \geq^{\#} j$.

7.1 Petri Net Markup Language

Instead of implementing yet another Petri net editor, I decided to use an existing net description language as input format to facilitate interoperability between the compiler and other Petri net tools. It appears that the *Petri net markup language* presented at <http://www.pnml.org> and in ISO/IEC 15909 was designed specifically for this rôle as an interchange format. It is an XML-based markup language partially derived from the language used by the Petri net kernel¹ project. It defines basic net elements, includes facilities for graphical layout information and can be extended to accommodate various Petri net extensions

¹<http://www2.informatik.hu-berlin.de/top/pnk/index.html>

and tool-specific annotations. While the standard contains many features utterly irrelevant to the compiler project, they only add insignificant complexity to the compiler. As free Petri net software – actually most academic projects – suffers seriously from fragmentation² the added effort is certainly worth it.

Instead of repeating an ISO-standard, the simple example of the PNML-file shown in Figure 7.1 will be discussed. Disregarding the XML-mandated version and namespace declarations, the first relevant information comes line 2 where the³ `net` starts and is given an `id`. A net might in principle be layouted across multiple `pages`, hence line 5, however this trivial example makes no use of it. Upon the page `places`, `transitions`, and `arcs` maybe put. The first two also have a `name`, places also an `initialMarking`. The various `graphics` informations informs displaying software such as an editor where to put the respective labels. Finally `toolspecific` annotations are also allowed, the editor used to produce this example⁴ uses this facility to store place capacities (or – in this case – the absence thereof). Finally `arcs` have a `source` and `target` specifying where they connect and carry an `inscription` which is customarily used to specify arc weight.

7.2 Architecture Overview

Converting such a PNML file to executable files is pretty straightforward: First a distribution function is constructed by collecting transitions connected by common preplaces. A C source file is then generated for each location. These sources are then compiled via any C-compiler (gcc was used in all experiments described later) to generate executables. Arcs incoming to a location are implemented by each location opening a listening TCP/IP socket where tokens can be deposited. Similarly, if a location has outgoing arcs, it connects to the sockets of the destination and deposits tokens when appropriate. This approach can implement arbitrary nets, however might generate only a single location if the net can not be subdivided further.

In principle, TCP/IP port numbers could be generated for each location, however that could easily lead to problems in deployment as unrelated services might also attempt to use the same ports. Hence listening port numbers are assigned by the operating system.

To ensure all locations still find each other, a central directory service is also generated where each location registers during startup and queries for the IP and port of all other locations it needs to know. This central directory service is only used during startup and only for address lookup. It could be removed by generating static addresses for all locations, however this would make network deployment a bit more hasslesome. An alternative would be some discovery protocol via broadcasts. As none of this has any impact on the distributed

²I really miss an efficient Petri net editor. It would certainly be worth the trouble, if it had more than one user.

³always singular

⁴<http://sourceforge.net/p/qtptneteditor/code/ci/master/tree/>

```

<?xml version="1.0" encoding="UTF-8"?>
<pnml xmlns="http://www.pnml.org/version-2009/grammar/pnml">
  <net id="net0"
    type="http://www.pnml.org/version-2009/grammar/ptnet">
5    <name><text>net0</text></name>
    <page id="page0">
      <name><text>net0</text></name>
      <place id="P0">
        <name>
10        <text>P0</text>
        <graphics><offset x="30" y="30"/></graphics>
        </name>
        <toolspecific tool="petrinet" version="1.0">
          <placeCapacity capacity="999999999"/>
15        </toolspecific>
        <graphics><position x="2379" y="2315"/></graphics>
        <initialMarking><text>1</text></initialMarking>
        </place>
        <place id="P1">
20        <name>
          <text>P1</text>
          <graphics><offset x="30" y="30"/></graphics>
          </name>
          <toolspecific tool="petrinet" version="1.0">
25          <placeCapacity capacity="999999999"/>
          </toolspecific>
          <graphics><position x="2379" y="2450"/></graphics>
          <initialMarking><text>0</text></initialMarking>
          </place>
30        <transition id="t0">
          <name>
            <text>t0</text>
            <graphics><offset x="30" y="10"/></graphics>
            </name>
35          <toolspecific tool="petrinet" version="1.0">
            <rotation degree="0"/>
            </toolspecific>
            <graphics><position x="2379" y="2392"/></graphics>
            </transition>
40          <arc id="a0" source="P0" target="t0">
            <inscription><text>1</text></inscription>
            </arc>
            <arc id="a1" source="t0" target="P1">
              <inscription><text>1</text></inscription>
45            </arc>
          </page>
        </net></pnml>

```

Figure 7.1: An example PNML file

execution of the implemented net however, the simple directory server solution seems straightforward and good enough.

Code generation is handled by a small perl script, which converts a PNML file into a directory containing the generated source files for all locations and the directory service together with a `Makefile` to compile all binaries and start them in parallel on the local machine for testing purposes.

The transformations of nets into more distributed versions presented in Figure 5.1 and Figure 5.4 have been implemented as separate perl scripts transforming PNML files to other PNML files. This gives the user the ability to inspect the resulting net⁵ before code generation.

7.3 Implementation Details and Networking

The network protocol is as simple as distributed network protocols get. The directory service starts first and at a known IP/port combination. Each location sends its own, automatically assigned, IP/port combination to the directory service and receives in return the array of the addresses of all other locations, possibly containing `INADDR_ANY` addresses entries where the respective location has not yet registered with the directory service. This repeats until none of the latter remain.

After this initial phase of node discovery, each location l creates a TCP/IP connection to all locations it might want to send tokens to, i.e. those targeted by arcs exiting from l . During compilation each place of the compiled net is assigned a unique id. When a location wants to deposit a token on a place, it sends a 4-byte integer⁶ along these connections. As all communication across location borders can happen asynchronously without ill effects, and TCP/IP will go to quite some lengths to deliver the tokens entrusted to its care, this protocol is sufficient.

While executing nets is a wonderful exercise in itself, the computer science practitioner will usually demand that his creations perform some visible, possibly useful, work. To this end the PNML to C transformation was augmented to take a further input file which associates to each label from Act a block of C code to execute upon firing of the transition(s) carrying said label. This facility was subsequently used to debug and benchmark the PNML to PNML transformations.

The code corresponding to a location l has to implement the firing rule for all transitions residing on l . To this end, it needs to decide of which transitions all preplaces hold a token, i.e. which are enabled. It then needs to choose one such transition for firing. While this disregards step semantics, any concrete actions associated with a transition would execute on the same thread anyway and step semantics would not be observable.⁷ After a transition is chosen,

⁵and me a decent way to debug the generation

⁶in host byte order

⁷If desired, ST-semantics can be implemented via a PNML to PNML transformation and judicious use of `fork` resp. `waitpid`

incoming tokens must be removed and outgoing tokens produced. The former amount to a trivial integer decrement, as the preplaces reside on l . The latter may involve sending tokens across the network.

The current number of tokens on a place is stored in a global integer array, which `enabled` and `fire` consult and modify. When a token is produced a function `sendToken` is invoked which consults a constant mapping from places to locations and either increments the local integer or issues a network write.

Most of the code, in particular everything pertaining to networking, is shared across locations by means of `#include` directives. The locations differ essentially in the implementation of the `enabled` and `fire` functions, which determine the enabledness of transitions and in the latter case also handle the token movement and the user-supplied code associated with transition labels.

Two strategies for selecting the transition to fire are implemented: One slightly faster, which considers the transitions in turn and fires the first to be found enabled. This approach will consistently decide conflicts the same way each time they occur, which is a correct semantics for non-determinism, but might run counter to some intuitions. The other strategy first collects all enabled transitions and then pseudo-randomly samples them uniformly for firing. Only the latter strategy guarantees progress in the case of a diverging net as after a transformation à la Figure 5.1.

These strategies are run until either no transition was found to be enabled, or a fixed upper number of firings occurred. After that the operating system is polled for network input, i. e. incoming tokens. These are then put into the target places and firing rule processing continues as before. While rather efficient, this of course increases the probability of firing “locally enabled” transitions relative to those which need remotely produced tokens. As the semantics considered in this thesis are not probabilistic, this does not pose a problem, however.

The implementation of the PNML to PNML transformations follows the formal definitions closely. To increase the probability of correctness, the actual \TeX source of the relevant figures was transformed into perl source. Each universal quantification resulted in an appropriate `foreach` loop. Exactly as in the formal setting, each net element was assigned a name upon generation. After all net elements are generated, they are unified using these names. Similarly, arcs are created based upon source and target name, thereby connecting the correct elements, even if arcs are generated in the scope of different quantifiers.

To enable a minimal degree of debuggability, graphical positions are assigned for generated elements based upon the positions of the original elements and the position of the element generator in the figure.

7.4 Performance and Comparison

To get an idea of the overhead introduced by the conflict replicating implementation, I applied it to the three test cases shown in Figures 7.2, 7.3, and 7.4.

Afterwards, each untransformed test case and each test case after applying the conflict replicating implementation was compiled and run using the tools

described in Section 7.2 in three variants:

- a) As a single executable without any distribution.
- b) As one executable per location, executed on a single machine M_1 .
- c) As one executable per location, executed on a “network” of two machines M_1 and M_2 . Four runs have been done using this setup to understand the effects of the (stochastic) allocation of locations to machines.

M_1 used an Dual Core, Intel Pentium CPU B960 clocked at 2.20GHz and had 8GB of RAM installed. The kernel version was `Linux version 3.8-1-amd64 (debian-kernel@lists.debian.org) (gcc version 4.7.2 (Debian 4.7.2-5)) #1 SMP Debian 3.8.11-1`. M_2 used a Quad Core, Intel Core2 Quad CPU Q8200 clocked at 2.33GHz and had 4GB of RAM installed. The kernel version was `Linux eta-carinae 3.11-trunk-amd64 #1 SMP Debian 3.11-1 exp1 (2013-09-12) x86_64 GNU/Linux`. The network between them was a consumer-grade 1 GBit/s Ethernet, with a round-trip time of 0.240ms⁸.

Naturally, the transformation to the conflict replicating implementation introduced new τ -transitions. In the case of Figure 7.2 approximately 23.3 transactions were required to simulate one original transition firing. For Figure 7.3 it took around 10.5 transitions to simulate one original firing. For Figure 7.4 it took between 26.8 and 28.4 implementation firings per original one, depending on the probability of various execution paths, which in turn depended on the concrete distribution of implementation locations across M_1/M_2 .

The Figures A.1ff report the concrete frequencies of transition firings for the various scenarios sampled over intervals of 1 second (and excluding the first 10 seconds to exclude the time when the distributed systems settle into equilibrium). The whiskers denote minimum and maximum numbers, the empty boxes range from first to third quartile, the line within the box denotes median.

One obvious thing to note is that compiling everything into a single executable is – as should be expected – vastly faster than any distributed variants. Remember though that this can only be the case because the transitions are not executing any time-using activities, as they might be in a more realistic system.

Further, it can be observed⁹ that some locations of the conflict replicating implementation are never fired at all. This suggests some optimisation potential, which is related to dead transitions being generated from Figure 5.4 which are not needed in some contexts.

Looking at the actual slowdown, when comparing the implementations of the largest test case of Figure 7.4, we find that the parallelism and communication introduced by the conflict replicating implementation incurs so much overhead that the total number of transition firings per second drops from $6.6 \cdot 10^4 T/s$ to $4.4 \cdot 10^4 T/s$ on a single machine and from $4.2 \cdot 10^4 T/s$ to $2.2 \cdot 10^4 T/s$ for (the average¹⁰ of) the two machine case. While not particularly great, I consider this

⁸average over 20 ICMP echo requests done via the `ping` tool

⁹not in the graphs maybe, but in the raw data

¹⁰over only four runs, so take the numbers with a grain of salt

a decent performance for a construction we essentially created as an existence proof.

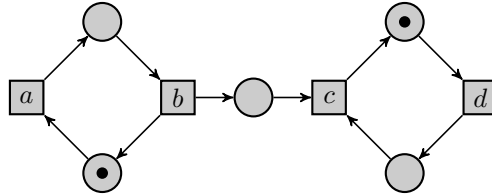


Figure 7.2: Testcase 1: Trivial producer-consumer example.

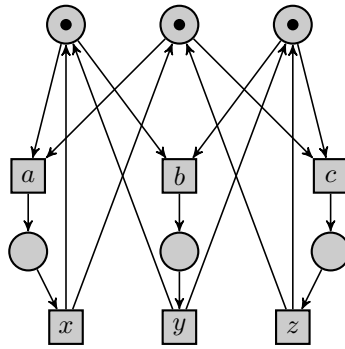


Figure 7.3: Testcase 2: A repeatedly firing, non-pure M.

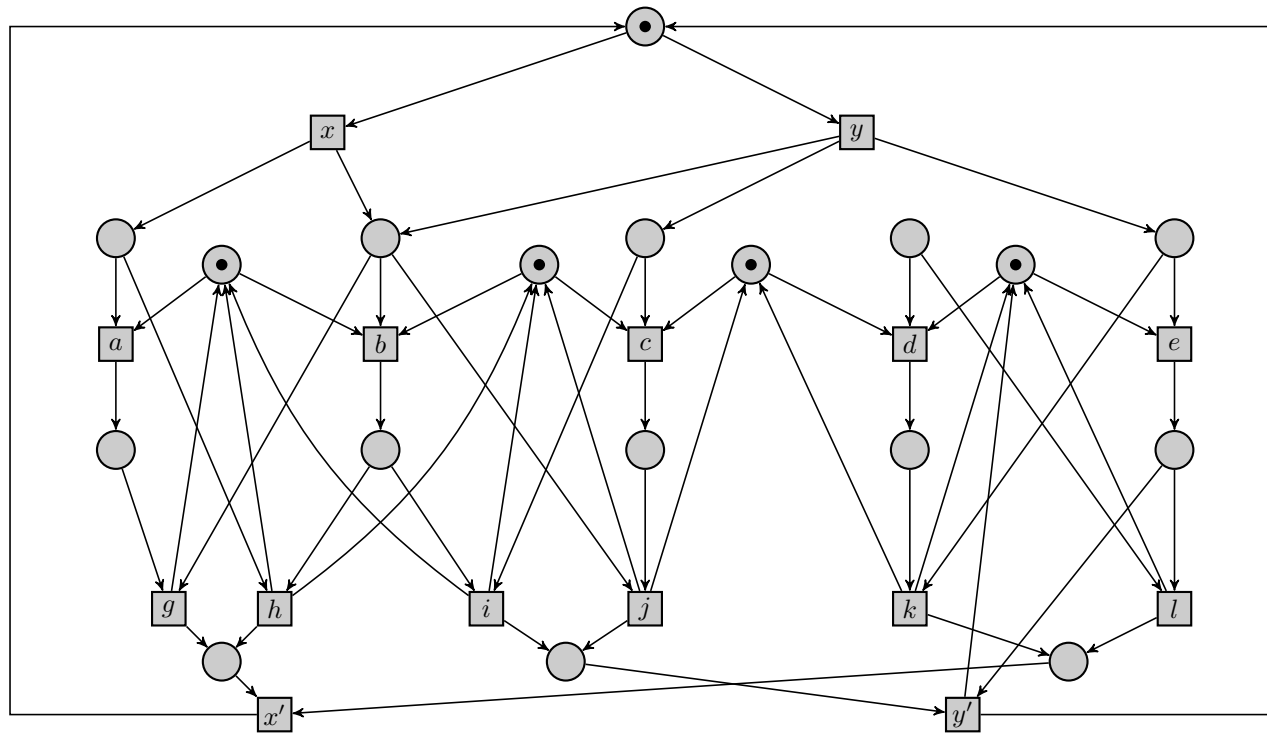


Figure 7.4: Testcase 3: A repeatedly firing structure consisting of multiple Ms.

Chapter 8

Context, Discussion and Conclusion

8.1 Ways to Evade the (Negative) Theorems

The preceding sections seem to make a clear case for avoiding Ms in distributed systems and abandoning any thoughts of distributed implementation whenever such a structure is encountered. Reality however is not quite as disagreeable. In particular, branching-time equivalence is often more than is needed. To begin with, many actions a system can take cannot be blocked by the environment in the first place. For many other actions, it is feasible to detect the current state of the environment before attempting the action or aborting it without ill effects if the environment is currently not permitting. Furthermore, relevant in particular when communicating with other systems, if a system consists of physically separate locations, it is highly likely that any single observer detecting its actions will do so only after the actions have been asynchronously communicated, again weakening the requirement of branching time correctness.

8.1.1 Probabilistic Algorithms

After the requirement of branching time has been loosened, the consensus problem remains, i.e. how to arrive at a consistent decision in all participating locations. Many kinds of consensus problems, including whether to execute b or $\{a, c\}$ in Figure 5.2 can be solved as follows: Each participant in the decision stochastically chooses a large integer and proposes which way the decision should go. The proposals together with the chosen integer are broadcast. The proposal associated with the largest number wins. If multiple participants have drawn the same number, depending on the requirements the process either repeats or the decision procedure aborts with an error. I strongly recommend the latter as it will highlight implementation problems more readily without compromising system reliability in production.

The probability of two participants choosing the same number can be made arbitrarily¹ small. The communication overhead can be reduced by aggregating sub-decisions instead of doing a full broadcast by each node. The main remaining problem is non-uniform sampling of the available integers, which might increase the probability of ties very considerably. This problem is particularly pronounced in embedded systems where real entropy is hard to come by but might also occur due to undetected programming errors anywhere in the software stack. Hence the above recommendation to error out, as a production system must deal with errors anyway, e.g. due to network problems, and just retrying to guess different random numbers might take an arbitrary number of rounds and thereby considerable time after entropy has been depleted.

A possible alternative is statically assigning unique location ids during deployment of the system and using them in lieu of random numbers. The downside is that any concrete process of assigning ids has its own probability of error which in practice is surprisingly large. The same problem naturally occurs when outsourcing the id assignment to a trusted third party, e.g. when using device identifiers or ethernet addresses as unique identification [Hir].

8.1.2 Time and Bounded Message Delay

Another assumption which can often be relied upon in practice but which is missing from nets as presented here is bounded message delay. In practice there is usually an upper bound after which a message will either have arrived or is otherwise assumed to be irretrievably lost. Nets as presented here however can neither detect the passage of “enough time” nor the absence of an expected token in a place. Hence certain solutions to distributed problems cannot be formulated in nets, respectively the version presented in this thesis. Extensions to nets exist which address both: [Ram74, MF76] introduced timed nets which can guarantee delivery of tokens after a finite amount of time, [AF73]² introduced inhibitor arcs to detect absence of tokens.

These two facilities are in particularly relevant to practice when dealing with failures, a problem the algorithms in this thesis do not burden themselves with.

Bounded message delay also becomes relevant when considering relaxed branching semantics in the sense that the environment can not block actions instantaneously but only after a known minimum time of advance warning as is oftentimes the case when considering embedded systems. Without bounded message delay a net cannot take advantage of such information as any attempt to communicate it across locations might take too long. If the message delivery bound is tight enough however, dissemination of information and a conflict resolution algorithm might be conducted while the information is still valid. This way further system specifications can be implemented distributedly in practice.

¹sufficiently small anyway [FS98]

²specifically, it was Agerwala who invented inhibitor arcs, as noted by his co-author in [Fly74] (prior to the publication of this thesis, “who invented inhibitor arcs” returned zero results in all search engines I tried).

Another aspect often overlooked in algorithm development, and likewise not pursued in this thesis, is the availability of global clocks. If used wisely, they can create a total ordering of events without communication between participants. The ordered events can later be communicated to other parts of the system and locally assembled into the same global history everywhere, resulting in a consistent view of the system state for each participant. While actual clocks are not strictly necessary for such a setup, they can make debugging and reasoning about a system a lot easier.

8.1.3 Quantum Mechanics

Many widely applied models of computing, including Petri nets, are based on classical physics, i. e. do not deal with quantum mechanical effects. Usually this poses not only no problem but restraints the solution space in a useful way. As already alluded to in some footnotes in the papers constituting this thesis, quantum mechanical effects might however come in handy when making consistent decisions in a spatially distributed system.

In particular they break arguments about irreducible symmetry, e. g. as made in [PN10, Pal03]. Consider a classical consensus problem: Two identical systems t and u wish to decide whether to execute an action at t or at u . However they are spatially separated. Whatever protocol you devise, as both systems are entirely identical, they can at the same time always make the same internal decisions, thereby always remaining in symmetrical internal states. Thus they will never come to a situation where t thinks that the other system should do the action, and u thinks that itself should do the action.

The obvious (and practically relevant) solution to the above problem is the stochastic technique already presented. It might however seem unsatisfactory on a theoretical level, as even physical randomness, say from thermal noise, cannot guarantee that the two systems diverge within a limited number of rounds. If such concerns bother you, consider the following (and entirely impractical) protocol based upon the Hong-Ou-Mandel effect [HOM87]: Let t and u each come with one part of a typical beam splitter, i. e. one triangular glass prism. The two systems are connected by the procedure of combining these two half-splitters into a working one. This setup is perfectly symmetrical. At the point in time t and u need to decide which one takes the action, they both send a photon into the beam splitter from their respective sides as in [BJD⁺06]. They then wait for photons coming out from the beam splitter to their respective side. If the two photons are indistinguishable, *either* t or u will observe any outcoming photons, thereby breaking the symmetry. If the photons are not indistinguishable, e. g. because they are not sent at the same time, the two systems would have diverged already beforehand, again solving the problem of symmetry.

Returning to the case of consistent decisions across a distributed system: In the general case or in the case of an M the environment makes possible and impossible the execution of certain transitions and this information must be transferred to remote locations before these can act upon it. Barring serious

disturbances of space-time[Alc94, Kra03] such transfer of information takes a positive, non-zero amount of time. Within this time however, the environment can alter its choice of allowed transitions. Hence the system can never act fast enough.

However, quantum entanglement can help with some cases where a consistent choice must be made between separated parts of the system, even though they cannot communicate and no viable pre-determined strategy exists. A concrete example was given in [BBT03] where (setting their $n = 3$) three players are challenged to output an even number of ones if all of them receive a zero (and not a one) as input and output an odd number of ones if exactly one player receives a zero.

Unfortunately, this challenge cannot be faithfully represented as a net in our framework as the complex constraints on input and output cannot be captured by an environment enabling and disabling individual transitions at will.

Nonetheless this shows that consistent system behaviour can be achieved without communication in a distributed system even in cases where classical models predict this to be impossible.

8.2 Related Work

Where possible, I referenced additional literature in the context where it is most relevant, i. e. in the preceding sections. Some works however are related to the contents of this thesis in a more general or abstract way – or simply had no place where they fit best. Such literature will be discussed in the following.

As before, most of the discussion can be found in the papers compiled into this thesis, i. e. [GGS08b, GGS09, GGSU11, GGS11b, GGS11a, GGSU13, MSUG14]

8.2.1 Petri Nets

We have not been the first to concern ourselves with the problem of distributed Petri net execution. The question whether and how a net can be implemented in a distributed way has been investigated at least in [Hop91, Tau88, BCD02, BD12]. In these works, given a distribution of the transitions of a net, the net is distributable iff it can be implemented by a net that is distributed w.r.t. that distribution. The requirement that concurrent transitions may not be co-located is absent; given the fixed distribution, there is no need for such a requirement. These papers differ from each other, and from our work, in what counts as a valid implementation.

The most similar work to our approach we know of is [Hop91], where Richard Hopkins introduces the concept of *distributable* Petri Nets. These are defined in terms of *locality functions*, which assign to every transition t a set of possible machines or locations $L(t)$ on which t may be executed, subject to the restriction that a set of transitions with a common preplace must share a common machine. A plain net N is distributable iff for every locality function L that can be

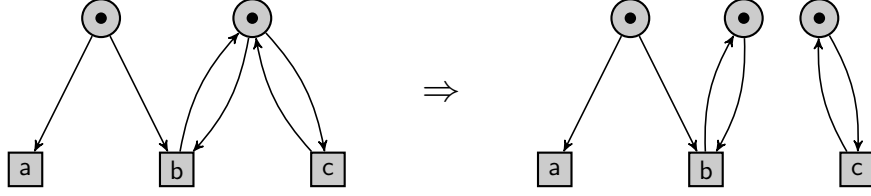


Figure 8.1: A specification and its Hopkins-implementation which added concurrency.

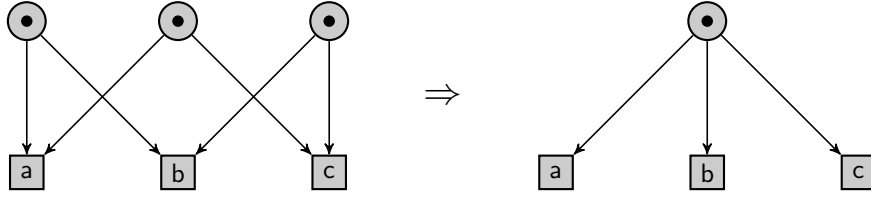


Figure 8.2: A distributable net which is not considered distributable in [Hop91], and our implementation of it.

imposed on it, it has a “distributed implementation”, a plain net N' with the same set of visible transitions, in which each transition is assigned a specific location, subject to three restrictions:

- the location of a visible transition t is chosen from $L(t)$,
- transitions with a common preplace must have the same location
- and there exists a weak bisimulation between N and N' , such that all τ -transitions involved in simulating a transition t from N reside on one of the locations $L(t)$.

The last clause enforces both a behavioural correspondence between N and N' and a structural one (through the requirement on locations). Thus, as in our work, the implementation is a τ -net that is required to be behaviourally equivalent to the original net. Hopkins allows implementations which are quite elaborate and make informed decisions based upon global knowledge of the net. As Hopkins notes, due to his use of interleaving semantics, his distributed implementations do not always display the same concurrent behaviour as the original nets, namely they add concurrency in some cases. This does not happen in our asynchronous implementations. Hence, the net classes he describes in his paper are incomparable with our class of distributable nets. One direction of this inequality depends on his choice of interleaving semantics, which allows the implementation in Figure 8.1. Already step failures equivalence does not tolerate the added concurrency and the depicted net is not distributable in our sense. The other direction of the inequality stems from the fact that we allow implementations which do not share structure with the specification but only emulate its behaviour. That way, the net in Figure 8.2 can be implemented in our approach as depicted.

The line of work from [Hop91] was continued at least in [BD12] where further research is done on the question of distributability when an assignment from transitions to locations is given. Multiple problematic patterns in Petri nets and the associated transition systems are newly identified, but the “exhaustive analysis of the induced situations” is still listed as an open problem.

Dirk Taubner has in [Tau88] given various protocols by which to implement arbitrary nets in the OCCAM programming language. Although this programming language offers synchronous communication he makes no substantial use of that feature in the protocols, thereby effectively providing an asynchronous implementation of nets. He does not indicate a specific equivalence relation, but is effectively using linear-time equivalences to compare implementations to the specification.

In [BCD02] an algorithm for the automated synthesis of distributed implementations of protocols is presented. The notion of distributed nets employed therein differs from ours by not requiring formally that no parallelism may occur on the same location. The authors however finally generate a finite automaton for each location, again serialising all actions on a single location. In contrast to our work and similar to [Hop91], the authors start with a user-supplied map from events to locations, and answer the concrete problem of whether that specific distribution is realisable or not instead of requiring the maximal possible parallelism to be realised.

A more abstract approach to the same underlying problem of correctly executing an arbitrary net as a distributed system has been taken in [KP13]. The authors provide a modified net semantics and an algorithm to split the net into agents which can locally decide most choices and resort to a global scheduler in case multiple agents must be coordinated. While such an approach loses branching time equivalence between a net and its implementation, it provides a clear separation of concerns between executing the net and solving the distributed coordination problems.

In [Sel97], Peter Selinger considers labelled transition systems whose visible actions are partitioned into input and output actions. He defines asynchronous implementations of such a system by composing it with in- and output queues, and then characterises the systems that are behaviourally equivalent to their asynchronous implementations. The main difference with our approach is that we focus on asynchrony within a system, whereas Selinger focuses on the asynchronous nature of the communications of a system with the outside world.

8.2.2 Process Algebra

While this thesis presents results on Petri nets only, our research project also considered process calculi, in particular the π -calculus, to help ensure that results would not just be artefacts of the chosen model. The results – achieved mainly by Uwe Nestmann and Kirstin Peters – were published in [PN10, PN12, Pet12, PNG13, PSN11]. The precise relation can be investigated by considering Petri net semantics for process algebras, as begun in [Men12].

Whereas on the Petri net side, we defined many concepts for asynchronous

nets anew, in the land of the π -calculus a hierarchy of calculi of varying degrees of asynchrony was already known, and provided π_{mix} , π_{sep} and π_a [HT91, Bou92] for further study. The core difference between these calculi is, in which ways communication and choice are allowed to interact.

In π_a , output actions are not allowed to guard a process different from \emptyset . In π_{sep} they may, but the prefixes guarding continuations after a choice must all be either input *or* output guards for each choice. Only in π_{mix} may input and output guards be mixed within the same choice.

The expressive power of mixed choice, i. e. choice arbitrarily mixed with test for input- and output communication availability (or guards in calculi parlance), in the π -calculus was analysed in [Nes00, Gor10, PN10, PN12, Pet12, PNG13], and before that already in [Pal03], whereas [PSN11] investigated encodings from π_{mix} into π_{sep} or π_a with respect to the preservation of causal independencies. As the main contribution of that paper, it was shown that – in the asynchronous π -calculus – there is a strong connection between synchronous interactions and causal dependencies. More precisely – analogue to the separation result on nets in Section 6.2 – that no encoding from the synchronous π -calculus with mixed choice into the asynchronous π -calculus preserves causal independence and satisfies all the criteria of [Gor10].

Ultimately we also want to validate applicable results in join calculus [FG96], where joint input [Nes98] is allowed to synchronise processes.

But also outside our research group, quite a number of papers consider the question of synchronous versus asynchronous interaction in the realm of process algebras and the π -calculus.

In [Bou88] Luc Bougé considers the problem of implementing symmetric leader election in the sub-languages of CSP obtained by allowing different forms of communication, combining input and output guards in guarded choice in different ways. He finds that the possibility of implementing leader election depends heavily on the structure of the communication graphs. Truly symmetric schemes are only possible in CSP with arbitrary input and output guards in choices.

In [BP91] Frank de Boer and Catuscia Palamidessi consider various dialects of CSP with differing degrees of asynchrony. In particular, they consider CSP without output guards and CSP without any communication based guards. They also consider explicitly asynchronous variants of CSP where output actions cannot block, i. e. asynchronous sending is assumed. Similar work is done for the π -calculus in [Pal97] by Catuscia Palamidessi, in [Nes00] by Uwe Nestmann and in [Gor06] by Daniele Gorla. A rich hierarchy of asynchronous π -calculi has been mapped out in these papers. Again mixed-choice plays a central role in the implementation of truly synchronous behaviour.

Just as we applied notions of behavioural equivalence to compare implementation and specification of nets, the question which translations between process calculi are “good” also needs an answer. The debate on this point is still ongoing in the literature. It is well known that there is a good encoding from the choice-free synchronous π -calculus into its asynchronous variant (see [Bou92, HT91, Hon92]). It is also well-known [Pal03, Gor10, PN10] that there

is no good encoding from π_{mix} into its asynchronous variant if the encoding translates the parallel operator homomorphically. Palamidessi was the first to point out that mixed choice strictly raises the absolute expressive power of the synchronous π -calculus compared to its asynchronous variant.

Analysing this result in [PN10], it was observed that it boils down to the fact that only the full π -calculus can break syntactic symmetries, whereas its asynchronous variant can not. Moreover, as already [Gor10] states, the condition of homomorphic translation of the parallel operator is rather strict. Therefore, Gorla proposes the weaker criterion of compositional translation of the source language operators. As proven by [PN12], this weakening of the structural condition on the encoding of the parallel operator turns the separation result into an encodability result, i.e. there is an encoding from π_{mix} into π_{a} variant with respect to the criteria of Gorla³. Analysing the encoding given by [PN12], it can be seen that it introduces additional causal dependencies.

8.2.3 Applications

The questions tackled in our theoretical research program have also been raised repeatedly in the fields of hard- and software design.

There are approaches to hardware design where asynchronous interaction is an intriguing feature due to performance and also security issues. Some examples can be found in the papers [Mis86], [Lam78] and [Lam03] for performance and [FML⁺03] for the security aspect. In [Lam03] Lamport considers arbitration in hardware and outlines various arbitration-free “wait/signal” registers. He notes that non-determinism is thought to require arbitration, but no proof is known. He concludes that only marked graphs can be implemented using these registers. Lamport then introduces “Or-Waiting”, i.e. waiting for any of two signals, but has no model available to characterise the resulting processes. The communication primitives used bear a striking similarity to our symmetrically asynchronous nets.

On the side of software engineering, *Message sequence charts* (MSCs), also contained in UML 2.0 under the name sequence diagrams, are a model for specifying interactions between components (*instances*) of a system.

In extended versions of MSCs or in live sequence charts (LSCs, see [HM03]), inline expressions allow the model to describe choices between possible behaviours in MSCs. Obviously, this requires some mechanism in order to make sure that the choice is performed in a coherent way across system components (see e.g. [GGW99] for a discussion of this type of problem). In Petri net semantics of such situations, we find a reachable \mathbf{N} , hence the net is not symmetrically asynchronously implementable. However, the net is \mathbf{M} -free.

A simple kind of MSCs are *basic message sequence charts* (BMCs) as defined in [Int96], where choices are not allowed. A Petri net semantics of BMCs with asynchronous communication and a unique sending and receiving

³This encoding is neither prompt nor is the assumed equivalence \asymp strict, so the separation results of [Gor10] do not apply here.

event for each message will yield nets with unbranched places (see for instance [GRG93]). Hence in this case the resulting nets are conflict-free and therefore fully asynchronously implementable.

Similarly, issues of consistent choice in the face of asynchronous communication have already been investigated in the context of distributed algorithms. Rabin and Lehmann observed in [RL94] that there is no fully symmetric distributed solution to the dining philosophers problem. Mazurkiewicz gives the converse in [Maz97], i.e. a distributed enumeration algorithm for all non-impossible cases of communication graphs.

From a business perspective, there is a growing consensus that concerns of availability and latency of a distributed system often trump the need for consistent choices (e.g. [Vog09]). As an alternative path, the algorithm from [Lam98] is employed and the added latency hidden behind various batching and caching layers [SVS⁺13]. One way or another, fundamental improvements in distributed algorithms have the potential to create a lot of value in this time of planet-wide distribution.

8.3 Conclusions and Open Questions

The development of this thesis has (again) confirmed some of the folklore of distributed system development: That distributed systems development is error-prone and subtle. We published an algorithm and a correctness proof of it in a peer-reviewed, specialist conference, and yet both the algorithm and the proof were wrong. Our error was ultimately revealed during testing I performed for this thesis.

Secondly, the experimental results also show – unsurprisingly – that communication overhead for synchronisation has massive performance implications, certainly so when communication cuts through various layers of caching and optimization. Practical applications of distributed algorithms must carefully weight these downsides against improvements in scalability.

At this point it should also be noted that no part of this thesis is concerned with handling component failures of any kind. Adding such facilities will complicate the algorithms further. As distributed systems by their very nature contain more components than non-distributed ones, the probability of *any* component failing is much higher for distributed systems. Thus, for practical applications, failure handling is a must.

The formal results of this thesis include

- two equivalent definitions of distributed systems in terms of Petri nets,
- an characterization of Ns and Ms as crucial structures preventing distribution,
- a concrete algorithm to maximally distribute all M-free nets, a proof of its correctness, and a prototypical implementation of it, also

- the other side of the argument that Ms constitute the exact boundary between systems which are distributable and those which are not, thereby
- the corollary that distributability under step branching time equivalences depends on run-time reachability properties and is thus undecidable in the general case, and that
- a hardware primitive which will make it possible to distribute arbitrary systems up to step branching time equivalences can not exist.

While the results above shed some light on the (in-)abilities of distributed systems in general, and on Petri net models of the same more specifically, some questions remain and some have found a more precise formulation.

To begin with, Ms have been shown to be stable with respect to step failures equivalence and finer equivalences and completed pomset trace equivalence and finer equivalences, but to be implementable in weak completed step trace and coarser equivalences. This opens the question, where exactly the boundary lies. More concretely, we have a long standing conjecture that there exists a behavioural equivalence which behaves like a branching time equivalence for such environments which can only interact with the system asynchronously and that Ms will *not* be stable under this equivalence. If true, this would yield distributed implementations of general Petri nets in many practical settings.

Our conflict replicating implementation also does not fully preserve the causal behaviour of nets. It is an open problem to find a class of nets that can be implemented distributedly while preserving divergence, branching time and causality in full.

The work on stability of Ms under causal equivalences so far only covers 1-safe nets. The proof of our conjecture, that Theorem 15 can be extended to non-safe nets is hence open. Also, further interesting structures and corresponding net classes might be identified in the spectrum of increasingly synchronous nets outlined in Section 6.3.

Regarding Ns, it is still open whether they are stable under linear time equivalences, say step trace equivalence.

On a higher level of applications, we expect our results to be useful for programming and modelling language design for distributed systems. Using a Petri net semantics of a suitable system description language, we could compare our class of distributed nets to the class of nets expressible in the language, especially when restricting the allowed communication patterns in the various ways considered in [BP91, Bou88] or in [Lyn96]. A first step in that direction has been taken with [PNG13]. Used the other way around, our results can be used as guidance when deciding which features to offer in a new language, as they show some constructs to be unimplementable in practice. Again, it would be helpful to develop formal mappings into existing language constructs which are more accessible to language developers.

Finally, recent advances in the understanding and practical applicability of quantum mechanics have undermined many assumptions in computing. This pertains also to our impossibility results. It would be a very interesting thing

in general to extend Petri nets to include quantum mechanical effects [\[Abr08\]](#). Maybe in such a way that places could not only hold (classical) tokens but be in a quantum state which is measured only when consumed by special measuring transitions.

Chapter 9

Literature

- [Abr08] Samson Abramsky. Petri nets, discrete physics, and distributed quantum computation. In *Concurrency, Graphs and Models*, pages 527–543. Springer, 2008. [doi:10.1007/978-3-540-68679-8_33](https://doi.org/10.1007/978-3-540-68679-8_33).
- [AF73] Tilak Agerwala and Michael J. Flynn. Comments on capabilities, limitations and “correctness” of Petri nets. *SIGARCH Computer Architecture News*, 2(4):81–86, December 1973. [doi:10.1145/633642.803973](https://doi.org/10.1145/633642.803973).
- [AKD98] Wil van der Aalst, Ekkart Kindler, and Jörg Desel. Beyond asymmetric choice: A note on some extensions. *Petri Net Newsletter*, 55:3–13, 1998.
- [Alc94] Miguel Alcubierre. The warp drive: hyper-fast travel within general relativity. *Classical and Quantum Gravity*, 11(5):L73, 1994. <http://arxiv.org/pdf/gr-qc/0009013.pdf>.
- [BBT03] Gilles Brassard, Anne Broadbent, and Alain Tapp. Multi-party pseudo-telepathy. In *Algorithms and Data Structures*, pages 1–11. Springer, 2003. <http://arxiv.org/abs/quant-ph/0306042>.
- [BCD02] Éric Badouel, Benoît Caillaud, and Philippe Darondeau. Distributing finite automata through Petri net synthesis. *Formal Aspects of Computing*, 13:447–470, 2002. [doi:10.1007/s001650200022](https://doi.org/10.1007/s001650200022).
- [BD87] Eike Best and Raymond Devillers. Sequential and concurrent behaviour in Petri net theory. *Theoretical Computer Science*, 55(1):87–136, 1987. See also: Eike Best and Raymond Devillers (1987): *Interleaving and Partial Orders in Concurrency: A Formal Comparison*. In M. Wirsing, editor: *Formal Description of Programming Concepts III*, 1987, pp. 299–321, North-Holland. [doi:10.1016/0304-3975\(87\)90090-9](https://doi.org/10.1016/0304-3975(87)90090-9).

- [BD12] Eike Best and Philippe Darondeau. Petri net distributability. In E.M. Clarke, I. Virbitskaite, and A. Voronkov, editors, *Perspectives of Systems Informatics - Revised Selected Papers presented at the 8th International Andrei Ershov Memorial Conference, PSI 2011, Novosibirsk*, volume 7162 of LNCS, pages 1–18. Springer, 2012. doi:[10.1007/978-3-642-29709-0_1](https://doi.org/10.1007/978-3-642-29709-0_1).
- [Bes87] Eike Best. Structure theory of Petri nets: The free choice hiatus. In W. Brauer, W. Reisig, and G. Rozenberg, editors, *Advances in Petri Nets 1986*, volume 254 of LNCS, pages 168–206. Springer, 1987. doi:[10.1007/978-3-540-47919-2_8](https://doi.org/10.1007/978-3-540-47919-2_8).
- [BHR84] Stephen Brookes, Tony Hoare, and Anthony Roscoe. A theory of communicating sequential processes. *Journal of the ACM*, 31(3):560–599, 1984. doi:[10.1145/828.833](https://doi.org/10.1145/828.833).
- [BJD⁺06] Jones Beugnon, Matthew P.A. Jones, Jos Dingjan, Benoît Darquié, Gaëtan Messin, Antoine Browaeys, and Philippe Grangier. Quantum interference between two single photons emitted by independently trapped atoms. *Nature*, 440(7085):779–782, 2006. doi:[10.1038/nature04628](https://doi.org/10.1038/nature04628).
- [BKO87] Jan Bergstra, Jan Willem Klop, and Ernst-Rüdiger Olderog. Failures without chaos: a new process semantics for fair abstraction. In M. Wirsing, editor, *Formal Description of Programming Concepts – III, Proceedings of the 3th IFIP WG 2.2 working conference*, Ebberup 1986, pages 77–103, Amsterdam, 1987. North-Holland.
- [Bou88] Luc Bougé. On the existence of symmetric algorithms to find leaders in networks of communicating sequential processes. *Acta Inf.*, 25(2):179–201, 1988. doi:[10.1007/BF00263584](https://doi.org/10.1007/BF00263584).
- [Bou92] Gérard Boudol. Asynchrony and the π -calculus. Rapport de Recherche 1702, INRIA Sophia-Antipolis, May 1992.
- [BP91] Frank S. de Boer and Catuscia Palamidessi. Embedding as a tool for language comparison: On the CSP hierarchy. In J.C.M. Baeten and J.F. Groote, editors, Proc. 2nd International Conference on *Concurrency Theory (CONCUR'91)*, Amsterdam, The Netherlands, volume 527 of LNCS, pages 127–141. Springer, 1991. doi:[10.1007/3-540-54430-5_85](https://doi.org/10.1007/3-540-54430-5_85).
- [BS83] Eike Best and Michael W. Shields. Some equivalence results for free choice nets and simple nets and on the periodicity of live free choice nets. In G. Ausiello and M. Protasi, editors, Proc. of the 8th Colloquium on *Trees in Algebra and Programming (CAAP '83)*, volume 159 of LNCS, pages 141–154. Springer, 1983. doi:[10.1007/3-540-12727-5_7](https://doi.org/10.1007/3-540-12727-5_7).

- [BW13] Eike Best and Harro Wimmel. Structure theory of petri nets. In K. Jensen, W. M. P. Aalst, G. Balbo, M. Koutny, and K. Wolf, editors, *Transactions on Petri Nets and Other Models of Concurrency VII*, volume 7480 of *LNCS*, pages 162–224. Springer, 2013. doi:[10.1007/978-3-642-38143-0_5](https://doi.org/10.1007/978-3-642-38143-0_5).
- [DE95] Jörg Desel and Javier Esparza. *Free Choice Petri Nets*. Cambridge University Press, New York, NY, USA, 1995.
- [EHH10] Dorsaf El Hog-Benzina, Serge Haddad, and Rolf Hennicker. Process refinement and asynchronous composition with modalities. In Natalia Sidorova and Alexander Serebrenik, editors, *Proceedings of the 2nd International Workshop on Abstractions for Petri Nets and Other Models of Concurrency (APNOC'10)*, Braga, Portugal, June 2010. <https://hal.inria.fr/hal-00779893>.
- [Esp98] Javier Esparza. Decidability and complexity of petri net problems – an introduction. In W. Reisig and G. Rozenberg, editors, *Lectures on Petri Nets I: Basic Models*, volume 1491 of *LNCS*, pages 374–428. Springer, 1998. doi:[10.1007/3-540-65306-6_20](https://doi.org/10.1007/3-540-65306-6_20).
- [FG96] Cédric Fournet and Georges Gonthier. The reflexive chemical abstract machine and the join-calculus. In *Proceedings of POPL '96*, pages 372–385. ACM, January 1996. doi:[10.1145/237721.237805](https://doi.org/10.1145/237721.237805).
- [Fly74] Michael J. Flynn. Studies in the organization of computer systems. Final report: June 15, 1970–December 31, 1974. Technical report, Johns Hopkins Univ., Baltimore, MD.(USA), 1974. doi:[10.2172/4242288](https://doi.org/10.2172/4242288).
- [FML⁺03] Jacques Fournier, Simon Moore, Huiyun Li, Robert Mullins, and George Taylor. Security evaluation of asynchronous circuits. In *Cryptographic Hardware and Embedded Systems-CHES 2003*, pages 137–151. Springer, 2003. doi:[10.1007/978-3-540-45238-6_12](https://doi.org/10.1007/978-3-540-45238-6_12).
- [FS98] Willy Fischler and Leonhard Susskind. Holography and cosmology. 1998. <http://arxiv.org/abs/hep-th/9806039>.
- [GG01] Rob J. van Glabbeek and Ursula Goltz. Refinement of actions and equivalence notions for concurrent systems. *Acta Inf.*, 37(4/5):229–327, 2001. doi:[10.1007/s002360000041](https://doi.org/10.1007/s002360000041).
- [GGS08a] Rob J. van Glabbeek, Ursula Goltz, and Jens-W. Schicke. On synchronous and asynchronous interaction in distributed systems. In E. Ochmański and J. Tyszkiewicz, editors, *Mathematical Foundations of Computer Science 2008*, volume 5162 of *LNCS*, pages 16–35. Springer, 2008. doi:[10.1007/978-3-540-85238-4_2](https://doi.org/10.1007/978-3-540-85238-4_2).

- [GGS08b] Rob J. van Glabbeek, Ursula Goltz, and Jens-W. Schicke. On synchronous and asynchronous interaction in distributed systems. Technical Report 2008-04, TU Braunschweig, 2008. Extended abstract in [GGS08a]. <http://arxiv.org/abs/0901.0048v1>.
- [GGS08c] Rob J. van Glabbeek, Ursula Goltz, and Jens-W. Schicke. Symmetric and asymmetric asynchronous interaction. Technical Report 2008-03, TU Braunschweig, 2008. Extended abstract in [GGS09].
- [GGS09] Rob J. van Glabbeek, Ursula Goltz, and Jens-W. Schicke. Symmetric and asymmetric asynchronous interaction. In *Proceedings of the First Interaction and Concurrency Experiences Workshop (ICE 2008)*, volume 229.3 of *ENTCS*, pages 77–95, Amsterdam, The Netherlands, The Netherlands, 2009. Elsevier. [doi:10.1016/j.entcs.2009.06.040](https://doi.org/10.1016/j.entcs.2009.06.040).
- [GGS11a] Rob J. van Glabbeek, Ursula Goltz, and Jens-W. Schicke. On causal semantics of petri nets. Technical Report 2011-06, TU Braunschweig, 2011. Extended abstract in [GGS11b].
- [GGS11b] Rob J. van Glabbeek, Ursula Goltz, and Jens-W. Schicke. On causal semantics of petri nets (extended abstract). In J.-P. Katoen and B. König, editors, *Proceedings 22nd International Conference on Concurrency Theory, CONCUR 2011, Aachen, Germany, September 2011*, volume 6901 of *LNCS*, pages 43–59. Springer, 2011. [doi:10.1007/978-3-642-23217-6_4](https://doi.org/10.1007/978-3-642-23217-6_4).
- [GGSU11] Rob J. van Glabbeek, Ursula Goltz, and Jens-W. Schicke-Uffmann. On distributability of petri nets. Informatik Bericht Nr. 2011-10, Institut für Programmierung und Reaktive Systeme, TU Braunschweig, 2011. Extended abstract in [GGSU12]. <http://arxiv.org/abs/1207.3597>.
- [GGSU12] Rob J. van Glabbeek, Ursula Goltz, and Jens-W. Schicke-Uffmann. *Foundations of Software Science and Computational Structures: 15th International Conference, FOSSACS 2012, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2012, Tallinn, Estonia, March 24 – April 1, 2012. Proceedings*, chapter On Distributability of Petri Nets, pages 331–345. Springer, 2012. [doi:10.1007/978-3-642-28729-9_22](https://doi.org/10.1007/978-3-642-28729-9_22).
- [GGSU13] Rob J. van Glabbeek, Ursula Goltz, and Jens-W. Schicke-Uffmann. On characterising distributability. *Logical Methods in Computer Science*, 9(3), 2013. [doi:10.2168/LMCS-9\(3:17\)2013](https://doi.org/10.2168/LMCS-9(3:17)2013).
- [GGW99] Thomas Gehrke, Ursula Goltz, and Heike Wehrheim. Zur semantischen Analyse der dynamischen Modelle von UML mit Petri-Netzen.

- In E. Schnieder, editor, *Proc. of The 6th Symposium on Development and Operation of Complex Automation Systems*, volume 6, pages 547–566, 1999.
- [Gla93] Rob J. van Glabbeek. The linear time – branching time spectrum II. In *Proceedings of the 4th International Conference on Concurrency Theory (CONCUR'93)*, pages 66–81, London, UK, 1993. Springer. doi:[10.1007/3-540-57208-2_6](https://doi.org/10.1007/3-540-57208-2_6).
- [Gla01] Rob J. van Glabbeek. The linear time – branching time spectrum I: The semantics of concrete, sequential processes. In J.A. Smolka, A. Bergstra, and S.A. Ponse, editors, *Handbook of Process Algebra*, pages 3 – 99. Elsevier, 2001. doi:[10.1016/B978-044482830-9/50019-9](https://doi.org/10.1016/B978-044482830-9/50019-9).
- [GLT09] Rob J. van Glabbeek, Bas Luttik, and Nikola Trčka. Branching bisimilarity with explicit divergence. *Fundamenta Informaticae*, 93(4):371–392, 2009. <http://arxiv.org/abs/0812.3068>.
- [Gor06] Daniele Gorla. On the relative expressive power of asynchronous communication primitives. In L. Aceto and A. Ingólfssdóttir, editors, *Proc. of 9th Intern. Conf. on Foundations of Software Science and Computation Structures (FoSSaCS '06)*, volume 3921 of LNCS, pages 47–62. Springer, 2006. doi:[10.1007/11690634_4](https://doi.org/10.1007/11690634_4).
- [Gor10] Daniele Gorla. Towards a Unified Approach to Encodability and Separation Results for Process Calculi. *Information and Computation*, 208(9):1031–1053, September 2010. doi:[10.1016/j.ic.2010.05.002](https://doi.org/10.1016/j.ic.2010.05.002).
- [GR83] Ursula Goltz and Wolfgang Reisig. The non-sequential behaviour of Petri nets. *Information and Control*, 57(2-3):125–147, 1983. doi:[10.1016/S0019-9958\(83\)80040-0](https://doi.org/10.1016/S0019-9958(83)80040-0).
- [GRG93] Peter Graubmann, Ekkart Rudolph, and Jens Grabowski. Towards a Petri net based semantics definition for message sequence charts. In *Proc. of the 6th SDL Forum (SDL '93)*, 1993.
- [GSW80] Hartmann J. Genrich and E. Stankiewicz-Wiechno. A dictionary of some basic notions of net theory. In W. Brauer, editor, *Advanced Course: Net Theory and Applications*, volume 84 of LNCS, pages 519–531. Springer, 1980. doi:[10.1007/3-540-10001-6_39](https://doi.org/10.1007/3-540-10001-6_39).
- [GV87] Rob J. van Glabbeek and Frits W. Vaandrager. Petri net models for algebraic theories of concurrency (extended abstract). In *Proc. PARLE '87*, volume 259 of LNCS, pages 224–242. Springer, 1987. doi:[10.1007/3-540-17945-3_13](https://doi.org/10.1007/3-540-17945-3_13).

- [GW89] R. J. van Glabbeek and W. Peter Weijland. Branching time and abstraction in bisimulation semantics (extended abstract). In G.X. Ritter, editor, *Information Processing 89*, Proceedings of the IFIP 11th World Computer Congress, San Francisco 1989, pages 613–618. North-Holland, 1989.
- [GW96] Rob J. van Glabbeek and W. Peter Weijland. Branching time and abstraction in bisimulation semantics. *Journal of the ACM*, 43(3):555–600, 1996. Extended abstract in [GW89]. [doi:10.1145/233551.233556](https://doi.org/10.1145/233551.233556).
- [Hir] Tatsushi Hiraki. Patent [US8089981](https://patents.google.com/patent/US8089981).
- [HM03] David Harel and Rami Marelly. *Come, Let's Play: Scenario-Based Programming Using LSC's and the Play-Engine*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2003.
- [HOM87] Chung Ki Hong, Zhe Yu Ou, and Leonhard Mandel. Measurement of subpicosecond time intervals between two photons by interference. *Physical Review Letters*, 59:2044–2046, November 1987. [doi:10.1103/PhysRevLett.59.2044](https://doi.org/10.1103/PhysRevLett.59.2044).
- [Hon92] Kohei Honda. Notes on Soundness of a Mapping from π -calculus to ν -calculus. With comments added in October 1993, May 1992.
- [Hop91] Richard P. Hopkins. Distributable nets. In *Advances in Petri Nets 1991*, volume 524 of LNCS, pages 161–187. Springer, 1991. [doi:10.1007/BFb0019974](https://doi.org/10.1007/BFb0019974).
- [HT91] Kohei Honda and Mario Tokoro. An Object Calculus for Asynchronous Communication. In *Proceedings of ECOOP '91*, volume 512 of LNCS, pages 133–147, 1991. [doi:10.1007/BFb0057019](https://doi.org/10.1007/BFb0057019).
- [Int96] International Telecommunication Union. Message sequence chart, 1996. Standard ITU-T Z.120.
- [Knu] Donald E. Knuth. FAQ on Knuth's personal homepage. <http://www-cs-faculty.stanford.edu/~uno/faq.html>.
- [KP13] Joost-Pieter Katoen and Doron Peled. Taming confusion for modeling and implementing probabilistic concurrent systems. In *Programming Languages and Systems - 22nd European Symposium on Programming, ESOP 2013, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2013, Rome, Italy, March 16-24, 2013. Proceedings*, volume 7792 of *Lecture Notes in Computer Science*, pages 411–430. Springer, 2013. [doi:10.1007/978-3-642-37036-6_23](https://doi.org/10.1007/978-3-642-37036-6_23).

- [Kra03] Serguei Krasnikov. Quantum inequalities do not forbid spacetime shortcuts. *Phys. Rev. D*, 67:104013, May 2003. [doi:10.1103/PhysRevD.67.104013](https://doi.org/10.1103/PhysRevD.67.104013).
- [Lam78] Leslie Lamport. Time, clocks, and the ordering of events in a distributed system. *Commun. ACM*, 21(7):558–565, 1978. [doi:10.1145/359545.359563](https://doi.org/10.1145/359545.359563).
- [Lam98] Leslie Lamport. The part-time parliament. *ACM Transactions on Computer Systems*, 16(2):133–169, 1998. [doi:10.1145/279227.279229](https://doi.org/10.1145/279227.279229).
- [Lam03] Leslie Lamport. Arbitration-free synchronization. *Distrib. Comput.*, 16(2-3):219–237, 2003. [doi:10.1007/s00446-002-0076-2](https://doi.org/10.1007/s00446-002-0076-2).
- [Lyn96] Nancy A. Lynch. *Distributed Algorithms*. Morgan Kaufmann Publishers, 1996.
- [Maz97] Antoni W. Mazurkiewicz. Distributed enumeration. *Information Processing Letters*, 61(5):233–239, 1997. [doi:10.1016/S0020-0190\(97\)00022-7](https://doi.org/10.1016/S0020-0190(97)00022-7).
- [Men12] Stephan Mennicke. An Operational Petri Net Semantics for the Join-Calculus. In B. Luttik and M. A. Reniers, editors, *Combined 19th International Workshop on Expressiveness in Concurrency and 9th Workshop on Structural Operational Semantics*, volume 89 of *Electronic Proceedings in Theoretical Computer Science*, pages 131–147, sep 2012. [doi:10.4204/EPTCS.89.10](https://doi.org/10.4204/EPTCS.89.10).
- [MF76] Philip M. Merlin and David J. Farber. Recoverability of communication protocols—implications of a theoretical study. *Communications, IEEE Transactions on*, 24(9):1036–1043, 1976. [doi:10.1109/TCOM.1976.1093424](https://doi.org/10.1109/TCOM.1976.1093424).
- [Mil89] Robin Milner. *Communication and Concurrency*. Prentice Hall International Series in Computer Science. 1989.
- [Mis86] Jayadev Misra. Axioms for memory access in asynchronous hardware systems. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 8(1):142–153, 1986. [doi:10.1145/5001.5007](https://doi.org/10.1145/5001.5007).
- [MSUG14] Stephan Mennicke, Jens-W. Schicke-Uffmann, and Ursula Goltz. On the step branching time closure of free-choice petri nets. In E. Abraham and C. Palamidessi, editors, *Formal Techniques for Distributed Objects, Components, and Systems*, volume 8461 of *Lecture Notes in Computer Science*, pages 232–248. Springer Berlin Heidelberg, 2014. [doi:10.1007/978-3-662-43613-4_15](https://doi.org/10.1007/978-3-662-43613-4_15).

- [Nes98] Uwe Nestmann. On the expressive power of joint input. In C. Palamidessi and I. Castellani, editors, *Proceedings of EXPRESS '98*, volume 16.2 of ENTCS. Elsevier Science Publishers, 1998. [doi:10.1016/S1571-0661\(04\)00123-9](https://doi.org/10.1016/S1571-0661(04)00123-9).
- [Nes00] Uwe Nestmann. What is a "Good" Encoding of Guarded Choice? *Information and Computation*, 156:287–319, 2000. [doi:10.1006/inco.1999.2822](https://doi.org/10.1006/inco.1999.2822).
- [OH86] Ernst-Rüdiger Olderog and Tony Hoare. Specification-oriented semantics for communicating processes. *Acta Informatica*, 23:9–66, 1986. [doi:10.1007/BF00268075](https://doi.org/10.1007/BF00268075).
- [Pal97] Catuscia Palamidessi. Comparing the expressive power of the synchronous and the asynchronous pi-calculus. In Conference Record of the 24th ACM SIGPLAN-SIGACT Symposium on *Principles of Programming Languages (POPL '97)*, pages 256–265. ACM Press, 1997. [doi:10.1145/263699.263731](https://doi.org/10.1145/263699.263731).
- [Pal03] Catuscia Palamidessi. Comparing the expressive power of the synchronous and the asynchronous π -calculus. *Mathematical Structures in Computer Science*, 13(5):685–719, 2003. [doi:10.1017/S0960129503004043](https://doi.org/10.1017/S0960129503004043).
- [Pet62] Carl Adam Petri. *Kommunikation mit Automaten*. Bonn: Institut für Instrumentelle Mathematik, Schriften des IIM Nr. 2, 1962.
- [Pet77] Carl Adam Petri. Non-sequential processes. GMD-ISF Report 77.05, GMD, 1977.
- [Pet12] Kirstin Peters. *Translational Expressiveness—Comparing Process Calculi using Encodings*. PhD thesis, TU Berlin, 2012. <http://nbn-resolving.de/urn:nbn:de:kobv:83-opus-37495>.
- [PN10] Kirstin Peters and Uwe Nestmann. Breaking symmetries. In S. B. Fröschle and F. D. Valencia, editors, *Proceedings of EXPRESS'10*, volume 41 of *EPTCS*, pages 136–150, 2010. [doi:10.4204/EPTCS.41.10](https://doi.org/10.4204/EPTCS.41.10).
- [PN12] Kirstin Peters and Uwe Nestmann. Is it a "Good" Encoding of Mixed Choice? In *Proceedings of FoSSaCS 2012*, volume 7213 of LNCS, pages 210–224, 2012. [doi:10.1007/978-3-642-28729-9_14](https://doi.org/10.1007/978-3-642-28729-9_14).
- [PNG13] Kirstin Peters, Uwe Nestmann, and Ursula Goltz. On Distributability in Process Calculi. In M. Felleisen and P. Gardner, editors, *Programming Languages and Systems - Proceedings 22nd European Symposium on Programming, ESOP 2013, Rome, Italy, March 2013*, volume 7792 of LNCS, pages 310–329. Springer, 2013. [doi:10.1007/978-3-642-37036-6_18](https://doi.org/10.1007/978-3-642-37036-6_18).

- [Pra85] Vaughan R. Pratt. The pomset model of parallel processes: Unifying the temporal and the spatial. In *Seminar on Concurrency, Carnegie-Mellon University*, pages 180–196, London, UK, 1985. Springer. doi:[10.1007/3-540-15670-4_9](https://doi.org/10.1007/3-540-15670-4_9).
- [PSN11] Kirstin Peters, Jens-W. Schicke, and Uwe Nestmann. Synchrony vs causality in the asynchronous pi-calculus. In B. Luttik and F. Valencia, editors, *Proceedings 18th International Workshop on Expressiveness in Concurrency*, Aachen, Germany, 5th September 2011, volume 64 of *Electronic Proceedings in Theoretical Computer Science*, pages 89–103, 2011. doi:[10.4204/EPTCS.64.7](https://doi.org/10.4204/EPTCS.64.7).
- [Ram74] Chander Ramchandani. Analysis of asynchronous concurrent systems by timed petri nets. Technical report, Massachusetts Institute of Technology, Cambridge, MA, USA, 1974.
- [Rei82] Wolfgang Reisig. Deterministic buffer synchronization of sequential processes. *Acta Inf.*, 18:115–134, 1982. doi:[10.1007/BF00264434](https://doi.org/10.1007/BF00264434).
- [Rei84] Wolfgang Reisig. Partial Order Semantics versus Interleaving Semantics for CSP-like Languages and its Impact on Fairness. In *Proc. of the 11th Colloquium on Automata, Languages and Programming*, pages 403–413, London, UK, 1984. Springer. doi:[10.1007/3-540-13345-3_37](https://doi.org/10.1007/3-540-13345-3_37).
- [RL94] Michael O. Rabin and Daniel J. Lehmann. On the advantages of free choice: A symmetric and fully distributed solution to the dining philosophers problem. In A. Roscoe, editor, *A Classical Mind: Essays in Honour of C.A.R. Hoare*, chapter 20, pages 333–352. Prentice Hall, 1994. An extended abstract appeared in *Proceedings of POPL’81*, pages 133–138, doi:[10.1145/567532.567547](https://doi.org/10.1145/567532.567547).
- [Ros98] Anthony Roscoe. *The Theory and Practice of Concurrency*. Prentice Hall, 1998. doi:[10.1007/3-540-63141-0_26](https://doi.org/10.1007/3-540-63141-0_26).
- [RT86] Grzegorz Rozenberg and Pazhamaneri Subramaniam Thiagarajan. Petri nets: Basic notions, structure, behaviour. In J.W. Bakker, W.-P. Roever, and G. Rozenberg, editors, *Current Trends in Concurrency*, volume 224 of *LNCS*, pages 585–668. Springer, 1986. doi:[10.1007/BFb0027048](https://doi.org/10.1007/BFb0027048).
- [Sch08] Jens-W. Schicke. *Studienarbeit: Asynchronous Petri Net Classes*, 2008. TU Braunschweig.
- [Sch09] Jens-W. Schicke. *Diplomarbeit: Synchrony and Asynchrony in Petri Nets*, 2009. TU Braunschweig.
- [Sel97] Peter Selinger. First-order axioms for asynchrony. In *Proc. 8th International Conference on Concurrency Theory (CONCUR’97)*,

- Warsaw, Poland, volume 1243 of LNCS, pages 376–390. Springer, 1997. [doi:10.1007/3-540-63141-0_26](https://doi.org/10.1007/3-540-63141-0_26).
- [SPG11] Jens-W. Schicke, Kirstin Peters, and Ursula Goltz. Synchrony vs. causality in asynchronous petri nets. In B. Luttik and F. Valencia, editors, Proceedings 18th International Workshop on *Expressiveness in Concurrency*, Aachen, Germany, 5th September 2011, volume 64 of *Electronic Proceedings in Theoretical Computer Science*, pages 119–131, 2011. [doi:10.4204/EPTCS.64.9](https://doi.org/10.4204/EPTCS.64.9).
- [SVS⁺13] Jeff Shute, Radek Vingralek, Bart Samwel, Ben Handy, Chad Whipkey, Eric Rollins, Mircea Oancea, Kyle Littlefield, David Menestrina, Stephan Ellner, et al. F1: A distributed SQL database that scales. *Proceedings of the VLDB Endowment*, 6(11):1068–1079, 2013. [doi:10.14778/2536222.2536232](https://doi.org/10.14778/2536222.2536232).
- [Tau88] Dirk Taubner. Zur verteilten Implementierung von Petrinetzen. *Informationstechnik*, 30(5):357–370, 1988. Technical report, TUM-I 8805, TU München. [doi:10.1524/itit.1988.30.5.357](https://doi.org/10.1524/itit.1988.30.5.357).
- [TV89] Dirk Taubner and Walter Vogler. Step failures semantics and a complete proof system. *Acta Inf.*, 27(2):125–156, 1989. [doi:10.1007/BF00265151](https://doi.org/10.1007/BF00265151).
- [VN82] Guy Vidal-Naquet. Deterministic languages of petri nets. In Claude Girault and Wolfgang Reisig, editors, *Application and Theory of Petri Nets*, volume 52 of *Informatik-Fachberichte*, pages 198–202. Springer Berlin Heidelberg, 1982. [doi:10.1007/978-3-642-68353-4_34](https://doi.org/10.1007/978-3-642-68353-4_34).
- [Vog93] Walter Vogler. Bisimulation and action refinement. *Theor. Comput. Sci.*, 114(1):173–200, 1993. [doi:10.1016/0304-3975\(93\)90157-0](https://doi.org/10.1016/0304-3975(93)90157-0).
- [Vog09] Werner Vogels. Eventually consistent. *Communications of the ACM*, 52(1):40–44, 2009. [doi:10.1145/1435417.1435432](https://doi.org/10.1145/1435417.1435432).

Kapitel 10

Deutschsprachige Zusammenfassung

Die wenigsten Computersysteme arbeiten heutzutage isoliert. Vielmehr interagieren sie üblicherweise mehr oder minder dauerhaft mit ihrer Umgebung. Gleichzeitig bestehen die meisten Systeme inzwischen aus mehreren Komponenten, die bei genauerer Betrachtung ebenfalls als eigenständige Computer verstanden werden können. Ganz offensichtlich gilt dies für Webanwendungen wie soziale Netzwerke, die Milliarden von Benutzern zur Verfügung stehen müssen. Aber auch auf der Skala einzelner Autos, Smartphones oder sogar medizinischer Implantate finden sich oft mehrere Teilsysteme.

Seit langem ist bekannt, dass verteilte Systeme schwer zu Entwerfen und zu implementieren sind. Viele der Probleme ergeben sich aus der Unfähigkeit selbst der besten Programmierer alle möglichen Systemabläufe eines solchen Systems zu überblicken – zu zahlreich sind die Varianten der zeitlichen Abläufe. Die nur in einzelnen Abläufen resultierenden Fehler sind leicht zu übersehen und teilweise sehr schwer zu verstehen. Grundsätzlichere Probleme ergeben sich aber sobald ein System konsistente Daten mehreren – typischerweise räumlich verteilten – Benutzern gleichzeitig zur Verfügung stellen muss oder mehrere Teile einer auch außerhalb des verteilten Systems verbundenen Umgebung steuern soll. In derartigen Kontexten können die Verzögerungen, die durch die Informationsweitergabe innerhalb des Systems entstehen, sichtbare Auswirkungen haben.

Die vorliegende Arbeit befasst sich mit genau diesen letzteren Problemen und stellt dar, welche Konsistenz in seinem Verhalten ein verteiltes System bestenfalls zeigen kann, wo Konsistenz nicht garantiert werden kann, und warum das so ist.

Um diesen Fragen nachzugehen, werden zuerst verteilte und asynchrone Systeme durch Petrinetze (im folgenden Netze) formalisiert und dann verschiedene verhaltensbasierte Äquivalenzrelationen zwischen Netzen definiert, die verschiedene Aspekte des Systemverhaltens unterschiedlich stark berücksichtigen. Durch die Auswahl der Äquivalenzrelation wird es so möglich, nur die für einen

bestimmten Anwendungsbereich relevanten Aspekte des Systemverhaltens zu betrachten.

Der Hauptteil der Arbeit klassifiziert dann, unter Verwendung dieser Formalisierung, Systeme bzw. Netze danach, ob ein verteiltes Netze existiert, dass sich äquivalent verhält. In einigen Fällen, in denen das nicht der Fall ist, zeigt die Arbeit, dass das Gegenbeispiel minimal in dem Sinne ist, dass alle anderen nicht verteilbaren Systeme das gegebene Beispiel als Teilnetz enthalten.

Anschließend wird eine ausführbare, wenn auch prototypische, Implementierung einiger der betrachteten Konstruktionen und der resultierenden Netze vorgestellt. Abschließend werden wie üblich verwandte Literatur und der größere Kontext diskutiert.

Die vorliegende Arbeit besteht größtenteils aus Papieren, die während der Jahre 2008 bis 2014 veröffentlicht wurden, wie jeweils zu Anfang der entsprechenden Kapitel und Abschnitte angegeben. Ich habe die Definitionen und Notation zwischen den Papieren größtenteils angeglichen, um den formalen Vergleich zu vereinfachen. Doppelte Inhalte habe ich natürlich ebenfalls zusammengefasst. Abschnitt [6.3](#), Kapitel [7](#) und Abschnitt [8.1](#) enthalten vollständig unveröffentlichtes Material.

Um das Verhalten zweier Netze zu vergleichen, gibt es zahllose Möglichkeiten [\[Gla93\]](#). In der vorliegenden Arbeit werden unter anderem [Step-Failures-Äquivalenz](#), [schwache Bisimilarität](#), [Branching-ST-Bisimilarität](#) und [Completed-Pomset-Trace-Äquivalenz](#) verwendet.

Step-Failures-Äquivalenz ist eine der größten Äquivalenzen, die sowohl die Entscheidungsstruktur als auch Nebenläufigkeit in einem System berücksichtigen. Schwache Bisimilarität berücksichtigt die Entscheidungsstruktur stärker und wird in der Literatur häufig verwendet. Branching-ST-Bisimilarität ist eine sehr feine Äquivalenz, im speziellen unterscheidet sie strikt feiner als Step-Failures-Äquivalenz. Completed-Pomset-Trace-Äquivalenz schließlich berücksichtigt die Entscheidungsstruktur eines Systems nicht, sondern nur die kausalen Abhängigkeiten zwischen Aktionen.

Die [Schaltregel](#) in [Petrinetzen](#) ist so definiert, dass beim Feuern einer Transition alle eingehenden Marken synchron entfernt werden. In verteilten Systeme ist dies jedoch eine unrealistische Annahme. In Abschnitt [3.1](#) wird daher jeder Stelle und jeder Transition eines Netzes ein Ort zugeordnet und ein synchrones Entfernen von Marken über Ortsgrenzen hinweg ausgeschlossen. Wären alle Netzelemente am selben Ort, wäre das System nicht verteilt. Dementsprechend werden anschließend Anforderungen an die Verteiltheit des Netzes gestellt und drei verschieden starke Varianten [solcher Anforderungen](#) vorgestellt: Volle Asynchronität, symmetrische Asynchronität und asymmetrische Asynchronität. Der Asynchronitätsbegriff ist somit zweifach parametrisiert: Zum einen darüber, welchen Grad der Asynchronität das verhaltensäquivalente Netz aufweisen muss, zum anderen welche Äquivalenzrelation genutzt wird, um das Netzverhalten zu vergleichen. Der Hauptteil der Arbeit befasst sich dann mit der Frage, welche Netzstrukturen für die verschiedenen Verteilbarkeitsvarianten dazu führen, dass

Netze verteilbar oder unverteilbar sind.

Als erste interessante Netzstruktur wird ein N identifiziert. In der Literatur existieren verschiedene Begriffe von free-choice Netzen, die diese Struktur mehr oder minder ausschließen. In der Arbeit wird unter anderem gezeigt, dass die Menge aller Netze, die schwach bisimilar zu einem symmetrisch asynchronen Netze sind, genau die Menge aller Netze ist, die schwach bisimilar zu einem free-choice Netz sind. Damit wird eine explizite, formale Begründung für die Definition von free-choice Netzen gegeben.

Bei gegebenen Anforderungen an ein System interessiert die exakte Umsetzung weniger. Daher werden in den weiteren Teilen der Arbeit statt der konkreten Asynchronitätsbegriffe das Konzept eines verteilbaren Netzes verwendet. Ein Netz ist dabei verteilbar, wenn irgendein verhaltensäquivalentes Netz existiert, das verteilt in dem Sinne ist, dass erstens alle Marken immer nur innerhalb eines Ortes entfernt werden, und zweitens niemals zwei Transitionen auf einem Ort im selben Schritt schalten. Dieser Verteilbarkeitsbegriff ist damit bezüglich der verwendeten Verhaltensäquivalenz parametrisiert.

Die zweite untersuchte Netzstruktur ist ein M. Es wird gezeigt, dass kein Netz, das ein M als Teilnetz enthält, verteilbar bezüglich Step-Failures-Äquivalenz ist. Selbiges gilt damit auch für alle feineren Äquivalenzen. Anschließend folgt ein konstruktiver Beweis, dass alle Netze ohne M verteilbar bezüglich Branching-ST-Bisimilarität mit expliziter Divergenz sind. Selbiges gilt damit auch für alle gröberen Äquivalenzen. Die beiden genannten Äquivalenzen umschließen einen großen Teil des Spektrums interessanter Äquivalenzen [Gla93], insbesondere im Bereich, der sog. Branching-Time-Äquivalenzen enthält. Für den umschlossenen Bereich zeigt diese Arbeit also, dass das M genau die Struktur ist, deren Anwesenheit bestimmt, ob ein Netz verteilbar ist.

Für größere Äquivalenzen hatte ich bereits in [Sch09] gezeigt, dass alle Netze verteilbar sind, solange Kausalität zwischen Aktionen keine Berücksichtigung in der Äquivalenz findet. In Abschnitt 6.2 zeige ich nun, dass sobald Completed-Pomset-Trace-Äquivalenz verwendet wird, also Kausalität berücksichtigt wird, wieder alle Netze mit M unverteilbar sind.

In unserer Modellierung gehen wir davon aus, dass ein M nicht verteilt realisierbar ist. Es stellt sich nun umgekehrt die Frage, was geschähe, wenn doch eine Möglichkeit gefunden würde (z.B. durch Ausnutzung quantenmechanischer Prozesse) ein alleinstehendes M zu verteilen. In Abschnitt 6.3 zeige ich, dass dies allein nicht ausreicht, um alle Netze verteilen zu können. Vielmehr lassen sich beliebig große Teilnetze finden, die nicht in einzelne Ms zerfallen und als ein Block verteilt werden müssten.

Da der Beweis in Abschnitt 5.1.3 sehr groß ist, erschien es mir ratsam, die angegebene Implementierung auch zu testen. Kapitel 7 beschreibt daher eine prototypische Implementierung der Konstruktion. Und in der Tat enthält die vorliegende Arbeit eine gegenüber der in [GGSU11] veröffentlichten Fassung korrigierte Konstruktion. Nebenbei zeigt die Implementierung auch, dass sogar eine triviale Übersetzung der aus unserer Konstruktion entspringenden Netze in ausführbare Programme eine für manche Bereiche ausreichende Performance aufweist.

Die Arbeit schließt in Kapitel [8](#) mit einer Übersicht über Kontext, Literatur und offene Fragen. Es enthält auch eine kurze Diskussion der Anwendbarkeit der Theoreme in der Praxis, insbesondere welche Möglichkeiten trotz der formalen Ergebnisse bestehen, reale Systeme die ein M enthalten, zu verteilen. Dies ist zum Beispiel möglich, in dem beliebig kleine Wahrscheinlichkeiten für Nicht-Termination in Kauf genommen werden. Unberücksichtigt ist in den Theoremen auch, dass üblicherweise robuste Annahmen über die relative Geschwindigkeit verteilter Uhren möglich sind. Außerdem basieren die meisten Systemmodelle – auch Petrinetze – derzeit auf klassischer Physik, berücksichtigen Quanteneffekte also nicht.

Appendix A

Experimental Data

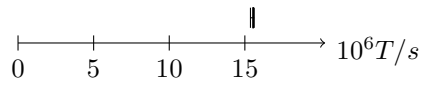


Figure A.1: Transactions per second, original net of Figure 7.2, single executable

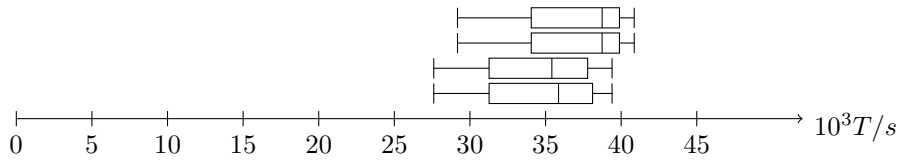


Figure A.2: Transactions per second and location, original net of Figure 7.2, single machine

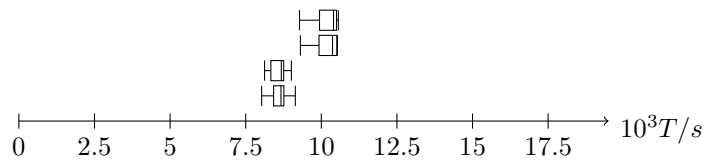


Figure A.3: Transactions per second and location, original net of Figure 7.2, two machines, run 1

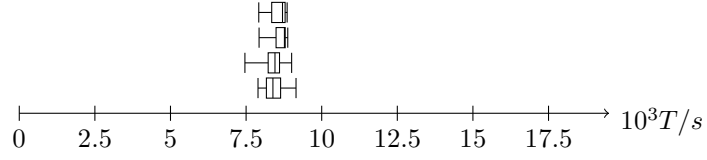


Figure A.4: Transactions per second and location, original net of Figure 7.2, two machines, run 2

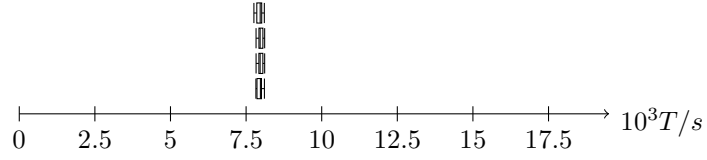


Figure A.5: Transactions per second and location, original net of Figure 7.2, two machines, run 3

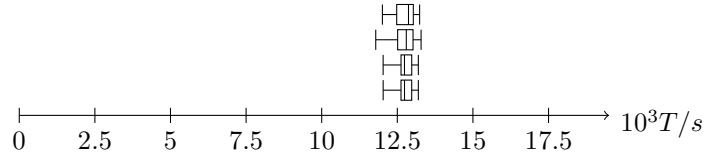


Figure A.6: Transactions per second and location, original net of Figure 7.2, two machines, run 4

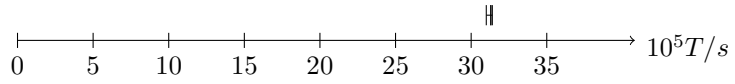


Figure A.7: Transactions per second, conflict replicating implementation of Figure 7.2, single executable

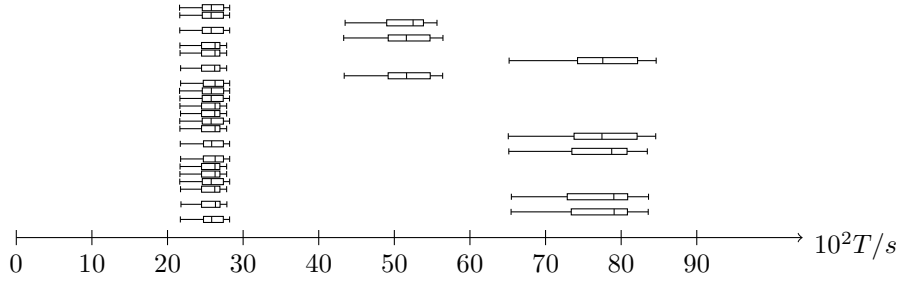


Figure A.8: Transactions per second, conflict replicating implementation of Figure 7.2, single machine

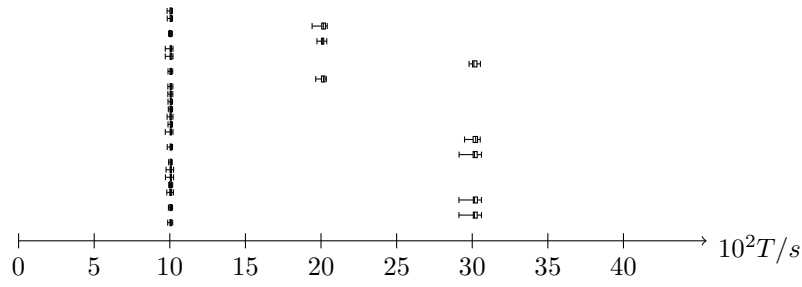


Figure A.9: Transactions per second and location, conflict replicating implementation of Figure 7.2, two machines, run 1

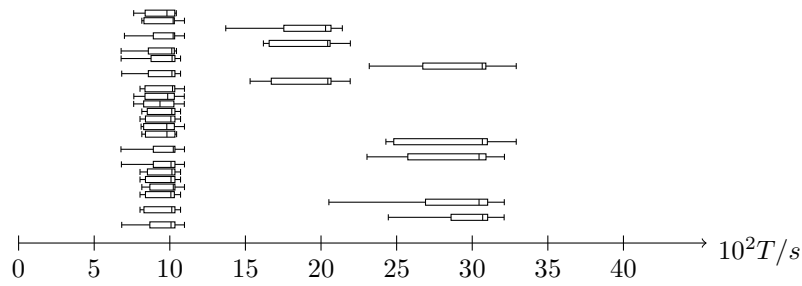


Figure A.10: Transactions per second and location, conflict replicating implementation of Figure 7.2, two machines, run 2

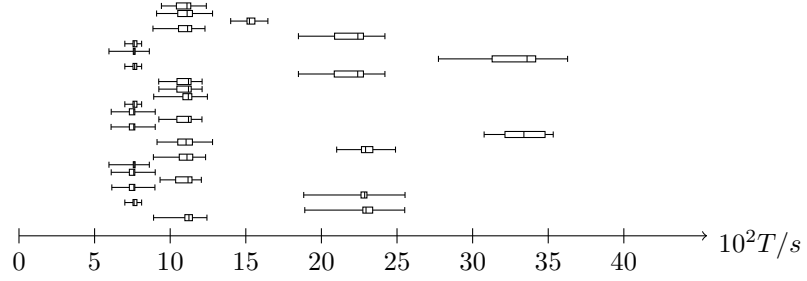


Figure A.11: Transactions per second and location, conflict replicating implementation of Figure 7.2, two machines, run 3

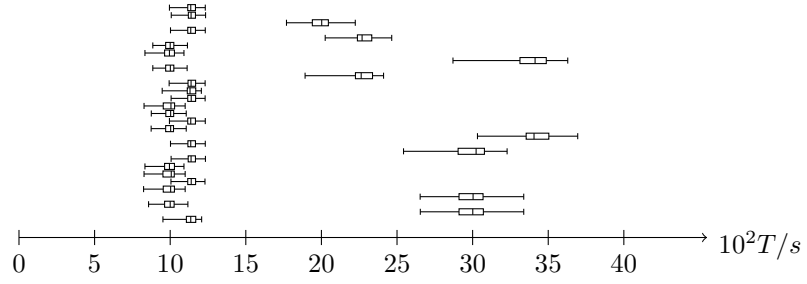


Figure A.12: Transactions per second and location, conflict replicating implementation of Figure 7.2, two machines, run 4

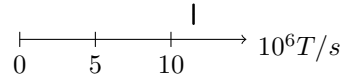


Figure A.13: Transactions per second, original net of Figure 7.3, single executable

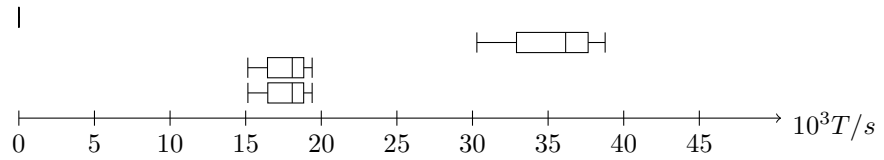


Figure A.14: Transactions per second and location, original net of Figure 7.3, single machine

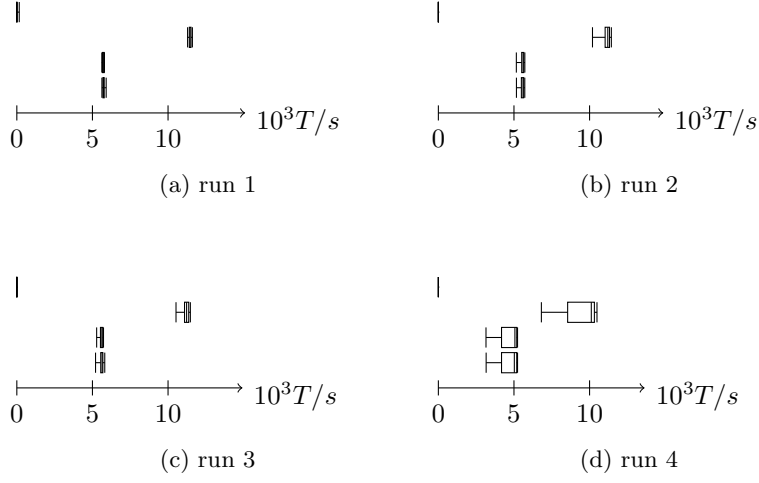


Figure A.15: Transactions per second and location, original net of Figure 7.3, two machines

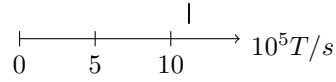


Figure A.16: Transactions per second, conflict replicating implementation of Figure 7.3, single executable

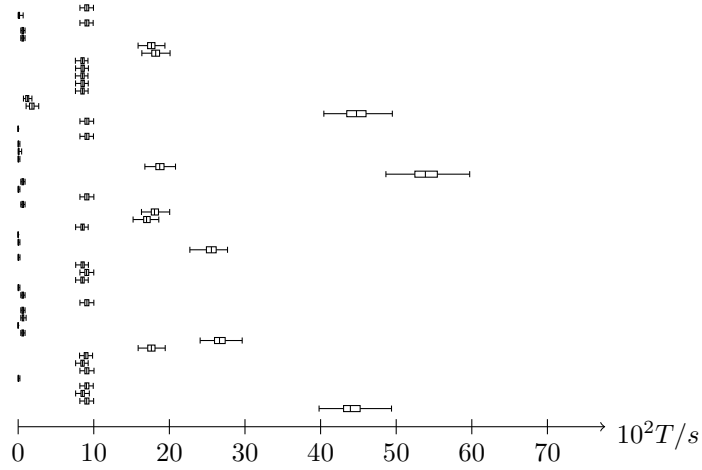


Figure A.17: Transactions per second and location, conflict replicating implementation of Figure 7.3, single machine

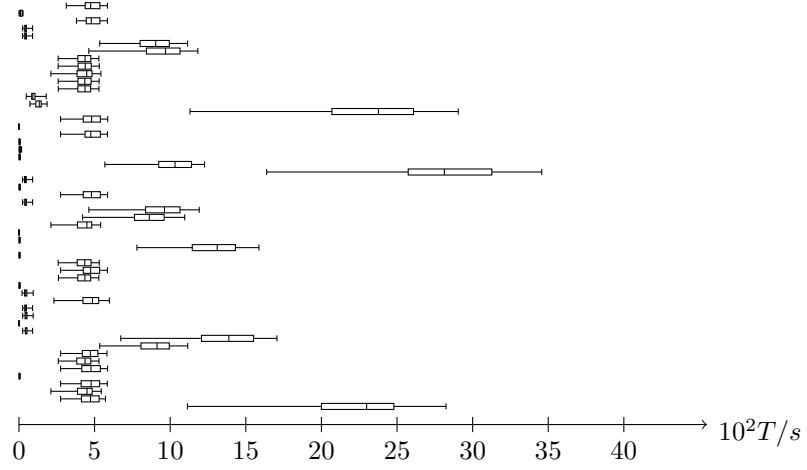


Figure A.18: Transactions per second and location, conflict replicating implementation of Figure 7.3, two machines, run 1

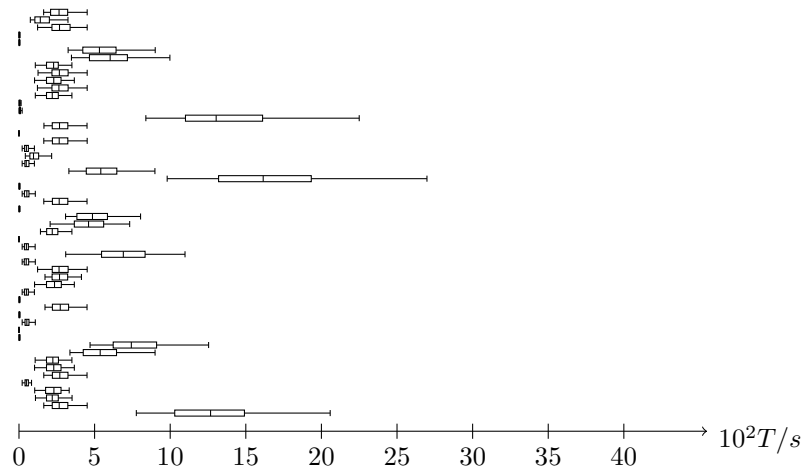


Figure A.19: Transactions per second and location, conflict replicating implementation of Figure 7.3, two machines, run 2

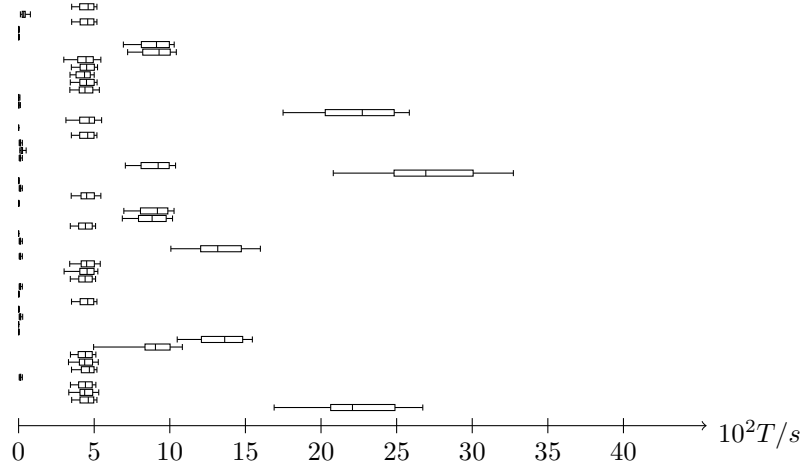


Figure A.20: Transactions per second and location, conflict replicating implementation of Figure 7.3, two machines, run 3

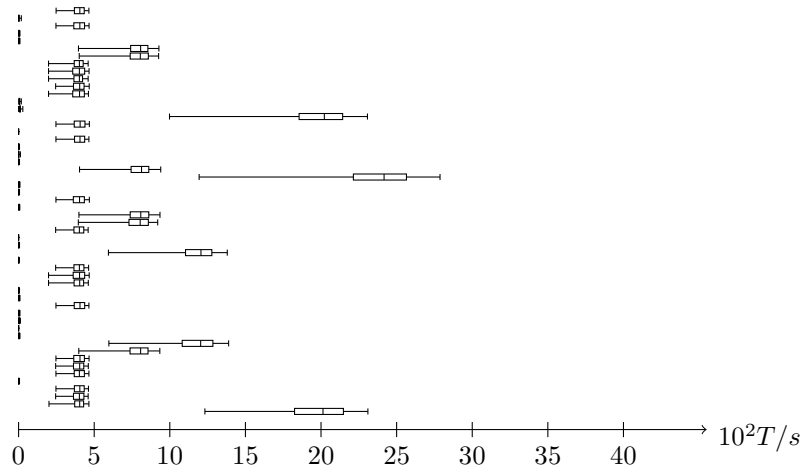


Figure A.21: Transactions per second and location, conflict replicating implementation of Figure 7.3, two machines, run 4

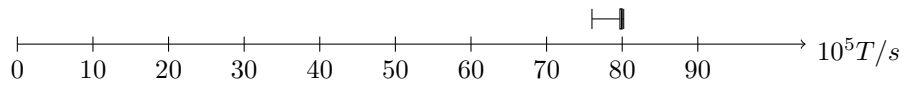


Figure A.22: Transactions per second, original net of Figure 7.4, single executable

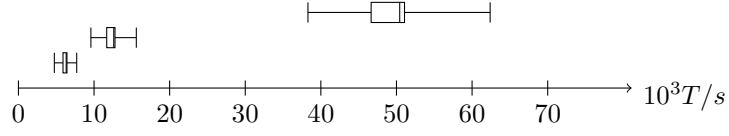


Figure A.23: Transactions per second and location, original net of Figure 7.4, single machine

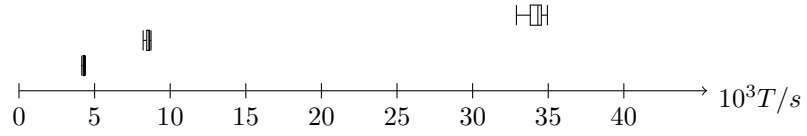


Figure A.24: Transactions per second and location, original net of Figure 7.4, two machines, run 1

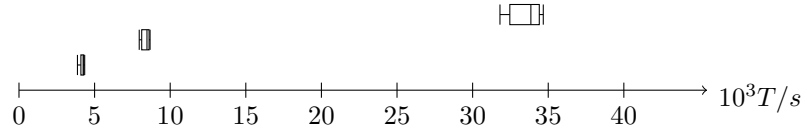


Figure A.25: Transactions per second and location, original net of Figure 7.4, two machines, run 2

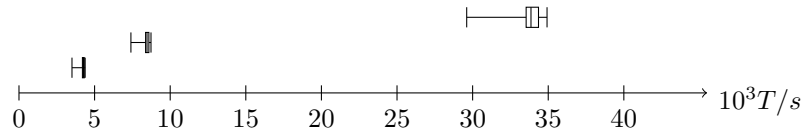


Figure A.26: Transactions per second and location, original net of Figure 7.4, two machines, run 3

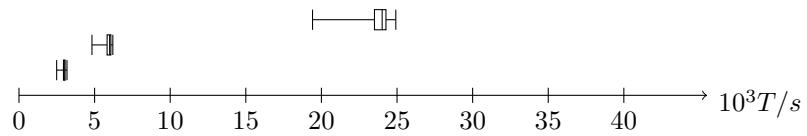


Figure A.27: Transactions per second and location, original net of Figure 7.4, two machines, run 4

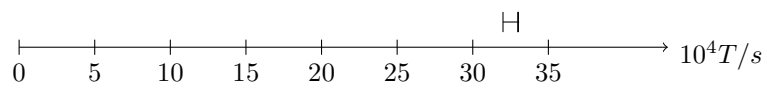


Figure A.28: Transactions per second, conflict replicating implementation of Figure [7.4](#), single executable

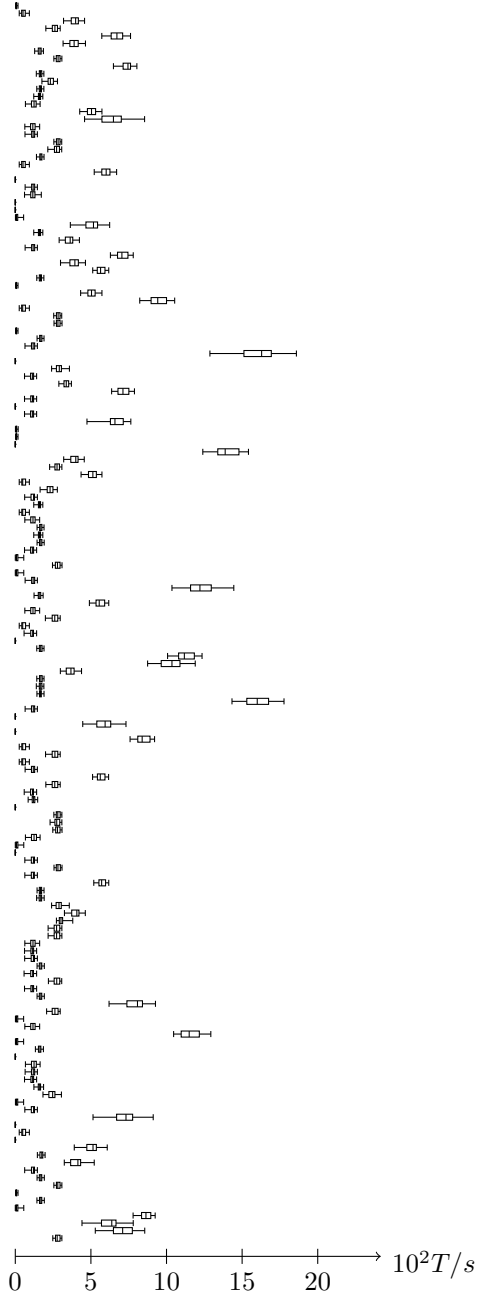


Figure A.29: Transactions per second and location, conflict replicating implementation of Figure 7.4, single machine

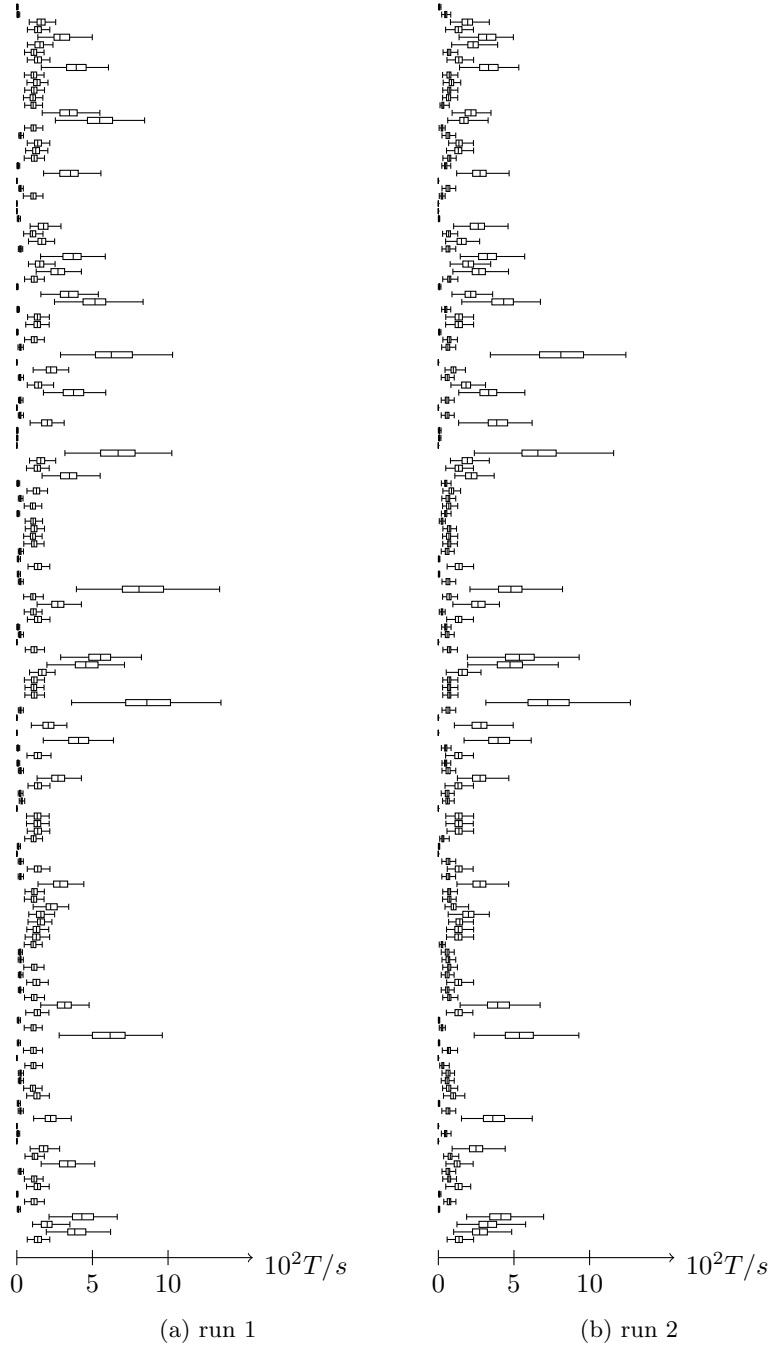


Figure A.30: Transactions per second and location, conflict replicating implementation of Figure 7.4, two machines

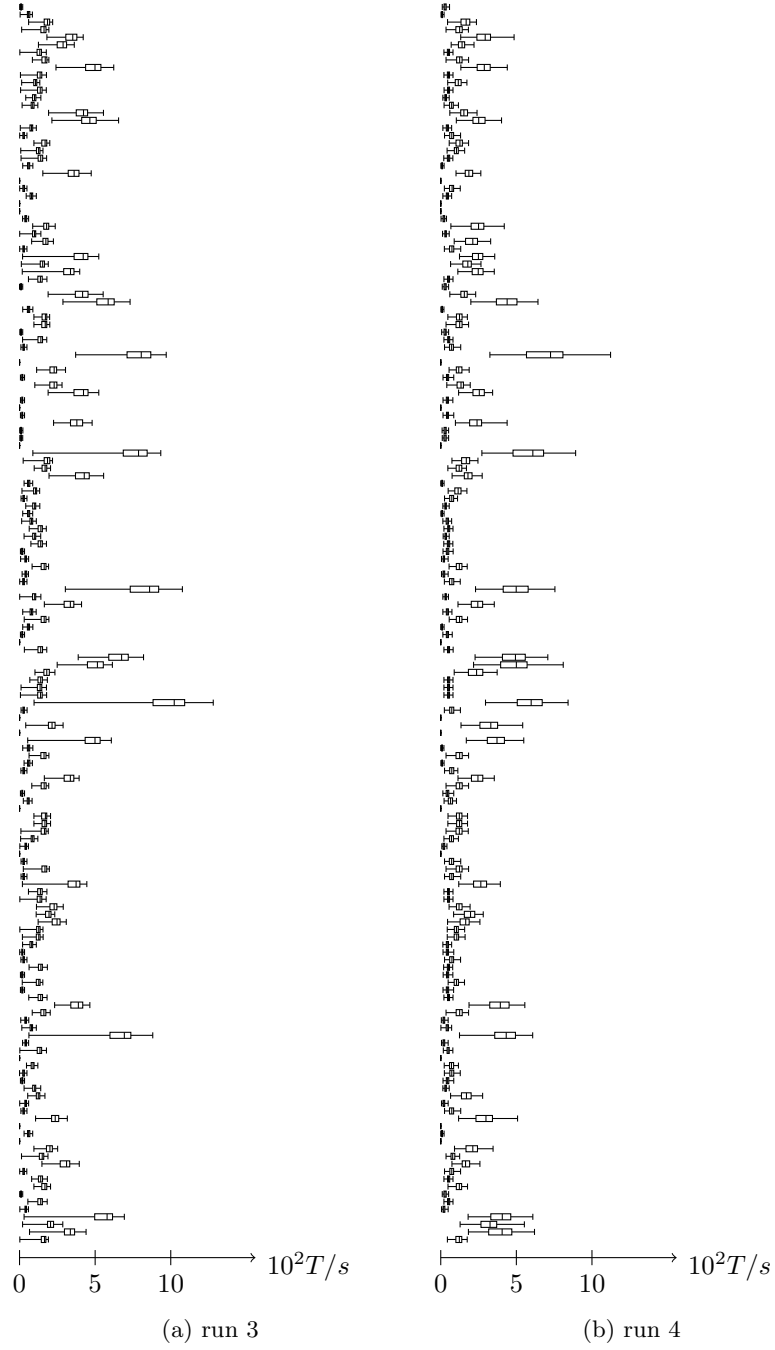


Figure A.31: Transactions per second and location, conflict replicating implementation of Figure 7.4, two machines

List of Figures

2.1 An example Petri net	12
2.2 A net with quantifiers.	13
2.3 The net of Fig. 2.2 expanded.	13
3.1 Possible results for $I_D(N)$ given different requirements	30
3.2 LSGA net from the essentially distributed net of Fig. 5.1.	45
3.3 Externally distributed, but not distributable net up to \approx_{bSTb}^Δ.	45
3.4 LSGA net from the externally distributed net of Fig. 3.3	45
4.1 The two net principles allowed in FC-nets.	47
4.2 Three nets that are not FC-nets.	48
4.3 Transformation to FC-nets and EFC-nets, respectively.	49
4.4 From BFC-nets to FC-nets.	52
4.5 Two AC-nets having a partially and fully reachable N.	54
4.6 Overview of results from Chapters 3 and 4.	59
5.1 A busy-wait implementation of the net in Fig. 5.2.	62
5.2 A fully reachable pure M.	64
5.3 A reversible transition and its macro expansion.	66
5.4 The entire conflict replicating implementation.	69
5.5 The entire conflict replicating implementation (macros expanded).	70
5.6 An example net.	72
5.7 The conflict replicating implementation of the net in Fig. 5.6	73
6.1 A repeated pure M. A finite, 1-safe, undistributable net.	108
6.2 An infinite implementation of Fig. 6.1.	109
6.3 The arthropod net of degree 4.	113
7.1 An example PNML file	119
7.2 Testcase 1: Trivial producer-consumer example.	123
7.3 Testcase 2: A repeatedly firing, non-pure M.	123
7.4 Testcase 3: A repeatedly firing structure of multiple Ms.	124
8.1 A net and its Hopkins-implementation which added concurrency.	129
8.2 A distributable net, but not considered distributable in [Hop91].	129

List of Tables

<u>5.1</u>	<u>Place/transition system of a net with reversible transitions.</u>	<u>. . .</u>	<u>67</u>
<u>5.2</u>	<u>The conflict replicating implementation.</u>	<u>.</u>	<u>88</u>